

Microsoft Win32 Developer's Reference Library, Volume 4:
Microsoft Windows Common Controls

(美) David Iseminger 主编
前导工作室 译



Win32 开发人员参考库

第4卷 Windows 通用控件



机械工业出版社
China Machine Press

Win32 开发人员参考库

第4卷 Windows 通用控件

套书介绍:

“Win32开发人员参考库”套书提供了Win32环境下的详尽编程指南, 能够帮助 Windows 开发人员有针对性地、高效而直观地获取自己所需的信息。本套书中的每卷都针对 Win32 编程环境的特定领域进行编写, 内容包括 Windows 基本服务、Windows 用户接口、Windows 图形设备接口、Windows 通用控件、Windows Shell。

本书介绍:

本书为开发人员充分利用 Windows 中大量预制的通用控件提供参考信息。对于每种通用控件, 书中都给出了此通用控件所对应的样式、通告消息、宏、函数以及相关的数据结构等详细信息。

- 《Win32 开发人员参考库 第1卷 Windows 基本服务》
- 《Win32 开发人员参考库 第2卷 Windows 用户接口》
- 《Win32 开发人员参考库 第3卷 Windows 图形设备接口》
- 《Win32 开发人员参考库 第4卷 Windows 通用控件》
- 《Win32 开发人员参考库 第5卷 Windows Shell》

适用水平: 中、高级

ISBN 7-111-08611-2



9 787111 086116



华章图书

www.china-pub.com

北京市西城区百万庄南街1号 100037

购书热线: (010)68995265, 8006100280 (北京地区)

ISBN 7-111-08611-2/TP · 1731

定价: 125.00 元

全套定价: 586.00 元

更多资源请访问稀酷客(www.ckook.com)

933

TP316.7

A19a

微软公司核心技术书库

Win32开发人员参考库

第4卷 Windows 通用控件

(美) David Iseminger 主编

前导工作室 译



A0983880



机械工业出版社
China Machine Press

更多资源请访问稀酷客(www.ckook.com)

本书介绍了Windows应用程序开发人员在整个开发过程中都可能会用到的通用控件，为开发人员充分利用Windows中大量预制的通用控件提供参考。内容包括各种通用控件的样式、通告消息、宏、函数以及相关数据结构。本书内容丰富、条理清晰，有效地提供了Windows通用控件信息。

David Iseminger: Microsoft Win32 Developer's Reference Library, Volume 4: Microsoft Windows Common Controls.

Copyright © 2001 by Microsoft Corporation.

Original English language edition copyright © 2000 by Microsoft Corporation; portions © 2000 by David Iseminger.

Published by arrangement with the original publisher, Microsoft Press, a division of Microsoft Corporation, Redmond, Washington, U.S.A. All rights reserved.

本书中文简体字版由美国微软出版社授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

本书版权登记号：图字：01-2001-0839

图书在版编目（CIP）数据

Win32开发人员参考库 第4卷 Windows通用控件/（美）艾塞明格（Iseminger, D.）主编；前导工作室译.-北京：机械工业出版社，2001

（微软公司核心技术书库）

书名原文：Microsoft Win32 Developer's Reference Library, Volume 4: Microsoft Windows Common Controls

ISBN 7-111-08611-2

I. W… II. ①艾… ②前… III. 窗口软件, Win32-程序设计 IV. TP316.7

中国版本图书馆CIP数据核字（2001）第00967号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：宋燕红

北京昌平第二印刷厂印刷·新华书店北京发行所发行

2001年4月第1版第1次印刷

787mm×1092mm 1/16·41印张

印数：0 001-5 000册

定价：125.00 元（全套586.00元）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

第一部分 概述

第1章 简介

欢迎使用“Win32开发人员参考库”套书，本套书是关于Win32开发环境的详尽参考指南。读者将会注意到，本套书中的每本书都针对各种技术或者开发概念进行了逻辑分组；采取这一方法是为了让读者（对于时间紧迫而信息超载的应用程序开发人员）可以更有针对性地来迅速、高效而直观地找到所需的信息。

除了着重提供Win32的参考资料之外，本套书还包括了来之不易的一些提示和诀窍，以使编程更为简单。例如，新版本的MSDN在线（MSDN Online）的全程解说和MSDN订阅（MSDN subscription）中的大多数帮助信息。读者还没订阅MSDN，或者不知道为何需要订阅吗？笔者也论述了这些信息，包括在三种级别MSDN订阅之间的差别，每种订阅可以提供什么，以及既然随时可通过Internet访问MSDN在线，为什么还要订阅MSDN。

Microsoft非常熟悉自身的编程技术，将其中一些知识共享不是很有意义吗？笔者认为很有意义的，这就是为何在本套书中读者可以找到这些知识的原因。本套书中每卷的第一部分中包含如何避免常见编程问题的建议。本套书的目标是成为读者关于Win32编程环境的一步到位的印刷版参考资源。

1.1 “Win32开发人员参考库”套书的结构

本套书包含5卷，每卷都针对Win32编程环境的某个特定领域进行讨论。5卷书如下：

- Win32开发人员参考库 第1卷 Windows基本服务
- Win32开发人员参考库 第2卷 Windows用户接口
- Win32开发人员参考库 第3卷 Windows图形设备接口
- Win32开发人员参考库 第4卷 Windows通用控件
- Win32开发人员参考库 第5卷 Windows Shell

5卷书是按功能分类的，让侧重于某些特定编程领域（比如用户接口）的软件开发人员，能够在特定的卷中获取特定的内容。通过这一途径，读者在查找Windows编程某一方面的内容时，随手打开某一本参考书籍即可。

在本套书的每一卷中，也设计了一个经过深思熟虑的结构。每卷的结构按照进一步强调对开发人员友好的原则来设计，让开发人员可以更容易地收集所需的信息。

第一部分介绍本套书以及本书各章节的编排信息。可以帮助读者了解Win32、MSDN和MSDN在线，包括一些提示和诀窍。

第二部分包含和本卷相关的Win32参考资料，但并不只是所定义的函数和结构的简单集合。

作为一个完备的参考资源，它在定义编程元素的同时，还包括了关于如何使用这些特定技术的信息。这些章节的信息包括完整的编程元素定义，以及每个编程领域的教材和说明资料。

1.2 “Win32开发人员参考库”套书的编写思想

本套书设计成易于接受的方式，来发布最为权威的信息。通过提供与电子版Microsoft参考信息一致的视图，本套书还设计成与MSDN和MSDN在线无缝集成。换句话说，对于在本书中出现的某给定函数的参考，也可以通过相同途径在MSDN和MSDN在线上得到对应函数的参考页面。

保持这种集成的原因很简单，为了让读者更易于使用这些工具来得到他们当前所需信息以创建高质量的程序。通过在参考资源中提供一个“通用界面”，如果读者已经熟悉本套书的参考资料，就可立即熟练使用MSDN和MSDN在线。总之，这是“一致性”带来的好处。

就任何工作而言，所使用的工具越简单越一致，就越会有更多的时间被花在工作上，而不是花在学会如何使用工具上。本套书的结构和编写思想，就是为了给读者提供一套完整而便利的工具，来构建出色的Windows应用程序。



第2章 本书中的内容

本书是“Win32开发人员参考库”套书的第4卷，主要介绍Windows应用程序开发人员在整个开发过程中都可能会用到的通用控件。本书为开发人员充分利用Windows中大量预制的通用控件提供参考材料。

在使用这些控件的过程中，开发人员必须正确处理与通用控件编程相关的版本控制问题。这里所讨论的所有控件几乎都包含在三个.dll文件（Comctl32.dll、Shell32.dll、Shlwapi.dll）中，并且这三个.dll文件都存在版本控制问题，必须在整个应用程序开发过程中对这些版本控制问题进行检查。Windows shell也存在与通用控件一样的版本控制问题；因此，无论是使用通用控件，还是使用Windows shell（在本套书第5卷中介绍），都必须处理版本控制问题。

那么，什么是版本控制问题呢？在本卷的第二部分（以及本套书的第5卷）对这些概念作了详细的说明，并且给出了进行版本控制所需掌握的所有信息。在参阅本卷中关于通用控件的详细说明，并且进行实际编程应用之前，应该先阅读第二部分中关于版本控制问题的相关内容。

在阅读了第二部分的开始内容，并且理解了在进行软件开发的过程中需要解决的版本控制问题之后，就可以进入第二部分中关于通用控件参考资料的学习。可以发现，通用控件列表很长，并且对某些控件的归类不一定很合理。但是，笔者并不是侧重于对这些控件进行归类，而仅仅提供关于这些通用控件的列表。幸运的是，许多通用控件名称自身就具有很强的自解释性。关于这些给定控件的更详细信息，请参见本书第二部分的内容并查找相应控件所在的章。

本书介绍的通用控件内容如下所示：

使用通用控件（第6章）

通用API（第7章）

定制控件外观（第8章）

动画控件（第9章）

扩展组合框控件（第10章）

创建向导（第11章）

日期与时间检出器控件（第12章）

拖放列表框（第13章）

平面滚动条（第14章）

头标控件（第15章）

热键控件（第16章）

IP地址控件（第17章）

月历控件（第18章）

Pager控件（第19章）

进度条控件（第20章）

属性页（第21章）

Rebar控件（第22章）

状态条（第23章）

选项卡控件（第24章）

工具提示控件（第25章）

轨迹条控件（第26章）

增减数控件（第27章）

本卷中的第二部分将对这些通用控件逐一说明。敬请读者记住，在实际学习使用这些通用控件之前，一定要阅读第二部分开头的简介内容，学习关于版本控制的问题。

第3章 使用Microsoft参考资源

在当今这个时代，问题不在于没有可用的信息，而在于信息的可用性。

不久前，要得到所需的信息很难，这是由于信息太少的缘故。为了找到所需的信息，用户需要找到可能有此信息的地点，然后亲自去访问这些地点，因为它并不在触手可及的地方或者在某个众所周知的重要位置上，所以这种搜索会耗费大量时间。简而言之，信息的可用性受到了限制。

今天，信息围绕着我们，有时信息多得令人窒息；我们面对太多的信息而显得负荷过重。如果我们不使用某些方法过滤掉那些不符合目标需求的信息，就会迅速被信息所淹没，看不清哪些是“垃圾信息”，哪些是有用的信息。概括地说，可用信息的超载使得我们找到真正所需的信息更为困难，泛滥成灾的信息减慢了我们的前进的速度。

这种情况也出现在Microsoft自身的参考资料上，并不是因为某些信息是不必要的，而是因为信息太多以至于定位所需信息变得很复杂。开发人员需要一条途径以剔除不相干的信息，获取所需内容。确保得到所需信息的一条途径，就是掌握工具的使用。木匠知道如何使用钉锤，会使他的工作更为高效；银行职员知道如何使用计算器，会使他的工作更为娴熟。如果用户是一位Windows应用程序的开发人员，就应该知道有两件工具：MSDN和MSDN在线。

“Win32开发人员参考库”这套书，针对给定Windows编程领域来提供参考资料。比较而言，MSDN和MSDN在线包含Microsoft在过去几年里收集的全部编程技术的所有参考资料，从而创建了一个信息量巨大的知识库。无论这些信息如何组织得当，但终究数量庞大，并且如果不知道如何下手；从中搜索所需内容（即使它就在手头）将会极其困难。

本章将向读者展示在MSDN和MSDN在线中导航所需的提示，让读者能够完全地发挥其能力。另外，在本章最后还分析了其他的Microsoft参考资源，用户将会掌握如何寻找所需的Microsoft参考信息（以及如何更快速和更高效地获取这些信息）。

3.1 Microsoft开发者网络

MSDN是Microsoft开发者网络（Microsoft Developer Network）的缩写，它的意图是向开发人员提供一个用以开发Windows应用程序的信息网络。许多人使用MSDN工作过或听说过MSDN，其中少数人有三种MSDN订购版之一，但是还有更多的人没有订阅MSDN，而在使用MSDN能为开发人员或者开发小组所做的某些简明指导。本章将深入介绍这些内容。

对于MSDN及其费用，还有一些需要说明的问题。如果用户听说过MSDN，或者有过MSDN在线的使用经历，就可能在使用这些资源的过程中询问以下问题：

- 如果类似于MSDN在线之类的资源，可以通过Internet免费访问到的话，为什么还需要订阅MSDN？

- 三种级别的MSDN订购版之间, 存在哪些差别?
- Site Builder Network有什么内容? 或者, Web Library是什么?
- MSDN与MSDN在线一个在CD上而另一个在Internet上, 除此之外, 它们有何不同? 它们的内容是重叠的、分离的、包含的还是其他关系?

如果用户提出这些问题, 这就意味着潜意识中对这个资源一直存在怀疑, 或者没有充分利用MSDN。

本章将介绍这些问题的全部答案, 以及关于如何最为高效地使用MSDN和MSDN在线的一些提示和帮助。

3.1.1 MSDN和MSDN在线的比较

MSDN和MSDN在线之间的差别之一, 就是在于哪一个拥有用户所需的特性。有时这一差别并不明显, 因为有些内容是通用的, 但是它们彼此之间是不同的。这些差别可以全部列出来吗? 可以, 以下是一般性的区别:

- MSDN提供参考内容和Microsoft最新的产品软件, 所有这些都使用CD形式发放给订阅人员 (在某些情况下使用DVD)。
- MSDN在线提供的参考内容和开发团体论坛, 都只可以通过Internet来访问。

微软公司通过多种途径让Windows开发人员得到他们所需的技术资料等信息, 而且微软公司的每一种分发机制都适合媒体的需要, 并且每种机制都以最好的方式来为“客户”提供可能的材料。这些在机制和媒体方面的考虑, 使得MSDN和MSDN在线可以提供给开发人员以多种不同的特性, 并且每种特性都有其优点。

MSDN比起MSDN在线来说, 可能不够“实时性”, 因为它使用邮件以CD的形式发放给订阅人员。但是, MSDN可以放入用户的CD驱动器 (或者放入硬盘中), 并且不受Internet速度或者网络失效的影响。另外, MSDN还有软件下载特性, 只要有最新的内容, 订阅人员就可以通过Internet自动更新本地MSDN目录, 而不用等待更新CD随邮件送到。MSDN显示界面——看起来像是定制的浏览器窗口——也和浏览器窗口一样可链接到Internet。为了协调MSDN和Internet的实时性, MSDN在线保留了部分站点给MSDN订阅人员, 允许一旦订阅材料可用即可同时进行更新 (在本地机器上)。

MSDN在线有大量可编辑和技术性的专栏, 这些专栏针对基于Windows Web站点或Windows应用程序的开发人员所遇到的问题、难点进行了适当裁剪。MSDN在线还有自定义的界面 (类似于MSN.com), 允许访问者根据最感兴趣的内容访问站点并进行处理。MSDN在线虽然具有最新的参考资料, 并且扩展了在线开发人员之间的交流但是它并不带有Microsoft产品软件, 并且不能够安装在本地机器中。

由于很容易辨别MSDN和MSDN在线之间的差别和类似之处, 因此可以很快指出它们侧重的区域。图3-1显示了MSDN和MSDN在线之间的差别以及类似之处。

一个需要注意的特性是, MSDN和MSDN在线之间的共同之处是界面, 它们的界面非常相似。这是为了确保开发人员使用MSDN的经验, 可以很容易地成为使用MSDN在线的经验, 反之亦然。

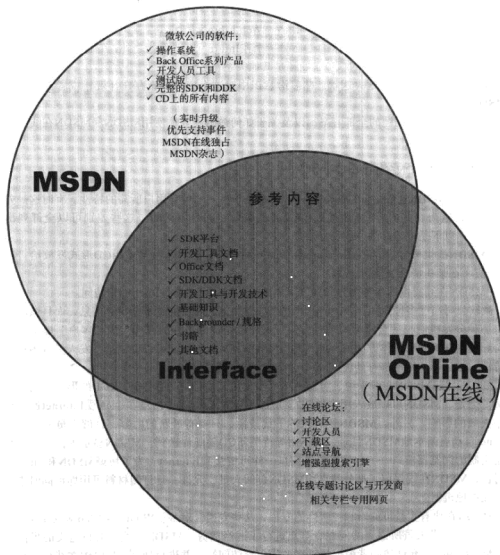


图3-1 MSDN覆盖范围和MSDN在线覆盖范围之间的相似与相异之处

还要记住的是，MSDN订阅人员仍然可以使用MSDN在线及其特性。如果有一份MSDN订购版，可能仍然需要继续使用MSDN在线，同时使用MSDN订购版提供的附加特性。

3.1.2 MSDN订购版

MSDN订购版有许多级别，本节介绍订购版级别的差别。

MSDN有三种订购版级别：Library（库）、Professional（专业）和Universal（通用）。每种都包含不同的特性。上一级别的订购版包含下一级别的所有特性，以及附加特性。换句话说，使用Professional订购版，人们得到在Library订购版中提供的所有特性，再加上附加特性；使用Universal订购版，可得到在Professional订购版中提供的所有特性，再加上附加特性。

1. MSDN Library订购版

MSDN Library订购版是基本的MSDN订购版本。虽然Library订购版不提供在Professional和Universal订购版中附带的Microsoft产品软件，但是它仍有开发人员在开发过程中有用的特性。使用Library订购版，用户可以得到下列内容：

- Microsoft参考库，包括SDK和DDK文档（每季度更新）。
- 大量示例代码，可供剪切和粘贴到用户项目中，完全免费。
- 完整的Microsoft知识库——漏洞和工作区的收集。
- Microsoft技术规范。
- 产品文档的完整系列，比如Visual Studio、Office以及其他软件。
- 某些书籍和杂志的完整（在某些情况下是部分的）电子版本。
- 讨论会和技术讲座论文——如果没有参加，就可以使用MSDN的记录。

除了这些内容之外，还可以得到：

- MSDN在线专栏的打包文件。
- 来自Microsoft支持的与开发相关的信息的定期E-mail。
- MSDN News的订阅，来自MSDN群体的双月刊报纸。
- 访问MSDN在线中订阅人员的专用区域和材料。

2. MSDN Professional订购版

MSDN Professional订购版是Library订购版的扩充。除了在前一小节所列出的特性之外，MSDN Professional订购版还提供如下内容：

- 完整的Windows操作系统集合，包括Windows 95/98/NT 4 Server/Workstation的发布版本。
- Windows SDK和DDK的完全内容。
- Windows操作系统的国际版本（可选）。
- 在开发和测试环境中的两次优先技术支持。

3. MSDN Universal订购版

MSDN Universal订购版是MSDN订购版中的最完备版本。除了在Professional订购版中提供的所有特性之外，MSDN Professional订购版还提供如下内容：

- 最新版本的Visual Studio，企业版。
- BackOffice测试平台，包括BackOffice家族中所有类型的Microsoft产品软件，每个软件都带有至少10个连接许可，以供用户软件产品开发之用。
- 附加的开发工具，比如Office Developer、Front Page和Project。
- 在开发和测试环境中的另外两次优先技术支持（总共四次）。

4. 购买MSDN订购版

当然，使用MSDN订购版所获得的所有特性都不是免费的。在写作本书时，MSDN订购版是

按年度进行订阅的。随着每种MSDN订购版在所包含功能特性上的逐步上升，价格也会逐步上升。请注意价格是波动的。

3.1.3 使用MSDN

MSDN订购版带有一个可安装的界面，并且Professional和Universal订购版还带有一批Microsoft产品软件，比如Windows平台版本和BackOffice应用程序。没有必要再向用户阐述如何使用Microsoft产品软件，但是还是要提供一些快速且有用的指导，以弄明白界面的含义，并在MSDN订购版所提供的似乎无穷无尽的参考资料中遨游。

对于使用过MSDN的用户来说，图3-2所示的界面是很熟悉的：它是MSDN参考资料的导航前台界面。

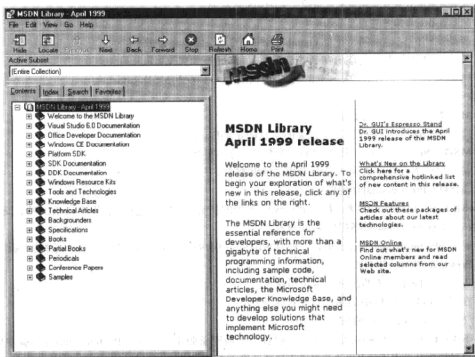


图3-2 MSDN界面

虽然界面很熟悉，并已经相当直观，但是如果还没有掌握其特性和浏览工具，可能还是会信息的汪洋大海中迷失方向的。通过掌握关于高效浏览的一些说明和提示，即可迅速提高其使用效率。

1. 浏览MSDN

MSDN最主要的特性之一（也许是最主要的缺点）是资料太完整，包含的信息总量超过1.1GB，并且还在不断增长。但是MSDN的创建者可能意识到了这一点，已经采取步骤缓解这一

问题。这些步骤之一就是允许开发人员有选择地在MSDN的目录中进行跳转。

MSDN的基本浏览很简单，与在Windows资源管理器及其文件夹结构中的浏览非常相似。MSDN没有使用文件夹，它将“书本”按照专题进行组织。点击书本左边的+号，可以将书本展开，显示出目录和嵌套的书本或者参考页面，如图3-3所示。如果在MSDN浏览器中没有看到左边的面板，请到View菜单再选择Navigation Tabs菜单项，面板就会出现。

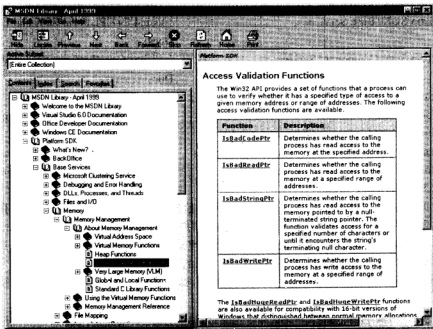


图3-3 MSDN浏览界面

MSDN左边面板中的四个选项卡是在MSDN目录中跳转的主要方式。这四个选项卡，与其上方的Active Subset下拉菜单框协同工作，是用以在MSDN目录中进行搜索的工具。在熟练掌握它们之后，这些浏览辅助工具将极大地丰富MSDN的使用经验。

Active Subset下拉菜单框提供的是一种过滤器机制。从下拉框中选择所感兴趣的MSDN信息子集，四个导航选项卡（包括Contents选项卡）都将显示的信息限制为所选子集中包含的信息。这就意味着在Search选项卡中所做的任何搜索，以及在Index选项卡中的索引，都被其定义的结果值过滤，并且/或者与用户所定义的子集相匹配，从而极大地限制了给定查询所得结果的数量，让用户可以得到真正需要的信息。在Index选项卡中，符合查询条件而不在于所选子集内，将被显示为浅色（但仍然可以选择），在Search选项卡中，它们将根本不会被显示。

MSDN中有下列预定义的子集：

Entire Collection

MSDN, Books and Periodicals

Platform SDK, Tools and Languages

Platform SDK, User Interface Services

MSDN,Content on Disk 2 only
 MSDN,Content on Disk 3 only
 MSDN,Knowledge Base
 MSDN,Office Development
 MSDN,Technical Articles and Backgrounders
 Platform SDK,BackOffice
 Platform SDK, Base Services

Platform SDK,Component Services

Platform SDK,Data Access Services
 Platform SDK,Graphics and Multimedia Services
 Platform SDK,Management Services
 Platform SDK,Messaging and Collaboration Services
 Platform SDK,Networking Services Platform
 SDK,Security

读者可以看到,过滤选项本质上是MSDN使用的信息结构的一个镜像。但是,如果用户希望看到这些子集的部分信息,又该怎么办呢?例如,用户对Platform SDK's Security、Networking Services和Management Services子集的某个关键字搜索以及Base Services子集的部分章节感兴趣,该如何去做呢?其实相当简单——仅需定义自己的子集即可。

选择View菜单,然后选择Define Subset菜单项,即可定义子集。用户面对的是如图3-4所示的窗口。

Platform SDK,Web Services
 Platform SDK,What's New?
 Platform SDK,Win32 API
 Repository 2.0 Documentation
 Visual Basic Documentation
 Visual C++ Documentation
 Visual C++,Platform SDK and Enterprise
 Docs
 Visual C++,Platform SDK,and WinCE
 Docs
 Visual FoxPro Documentation
 Visual InterDev Documentation
 Visual J++ Documentation
 Visual SourceSafe Documentation
 Visual Studio Product Documentation

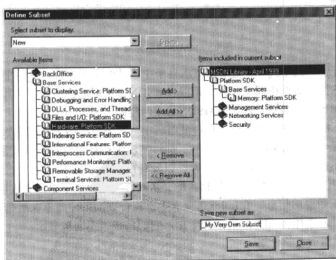


图3-4 Define Subset窗口

定义子集是很简单的，仅需按照如下步骤进行：

1) 选择希望放置到新子集的信息；可以选择整个子集，或者在可用子集中选取“书本”/“目录”。

2) 点击Add按钮，将选取的信息放入新创建的子集。

3) 在Save New Subset As文本框中键入名称，为新创建的子集命名。注意所定义子集（包括用户自己创建的）将以字母表顺序排列。

如果愿意的话，也可以从MSDN安装中删除整个子集。仅需在Select Subset To Display下拉框中选择希望删除的子集，然后点击旁边的Delete按钮即可。

一旦定义了一个子集，它将和预定义子集一样在MSDN中可用。并且在四个导航选项卡中也会过滤可用信息，和预定义子集表现相同。

2. 快速提示

现在读者知道如何浏览MSDN了，还有一些提示和诀窍，可以利用它们来尽可能高效地使用MSDN。

- 使用Locate按钮定位 也许是人的天性，需要知道自己身处何方。但遗憾的是，如果在右边面板显示参考页面（也许是从搜索结果中跳转过来的），而对于左边面板的Contents选项卡，没有与信息树中同步地显示参考页面的位置，会很让人头疼。即使用户知道参考页面所在的技术目录，但最好还是能够找出在目录结构中的位置。此问题解决起来很容易：仅需点击导航工具条中的Locate按钮，所有的参考将会同步。
- 像浏览器那样使用Back按钮 导航工具条中Back按钮的功能和浏览器Back按钮的一样。如果需要先前浏览过的某个参考页面信息，就可以使用Back按钮返回该页，而不用再重新经历一次搜索过程。
- 定义用户子集并且使用它们 如本章开始时所述，当前可用的信息量太多，有时让人无所适从。定义MSDN的子集，可以筛选出和工作相关的内容，使得工作起来效率更高。
- 在命名子集的开始使用下划线 在Active Subset下拉框里的子集是按照字母表进行排列的，而下拉框每次只能显示少量子集（使得选中可用子集比较困难）。下划线在字母表中位于字母之前，因此如果在所有自定义子集的名称都以下划线开始的话，它们就会在Active Subset下拉框中处在可用子集的前面。另外，使用下划线，用户可以一眼就看出哪些子集是自定义的，而哪些子集是MSDN附带的——这至少可以节约几秒钟，而这些几秒钟是可以累加的。

3.1.4 使用MSDN在线

MSDN在线和MSDN有许多类似之处，这并不意外。当用户从一个开发人员资源跳转到另一个时，如果能够立即使用该资源，那么他的工作将会变得更为轻松。但是，MSDN在线还是有相当多的差异，需要有所说明。相对MSDN而言，它具有不同的分发媒体，可以利用Internet这一途径，而MSDN却不能。

如果读者在此之前使用过Microsoft的主页（www.msn.com或者home.microsoft.com），就会

对定制页面不感到陌生。用户可以从可用的美国国内新闻、计算机新闻、本地新闻和天气、股票价格以及其他信息或新闻等分类中,选择自己感兴趣的内容。用户甚至还可以加入一些Web链接,使其在访问站点时可用。MSDN在线的主页也可以按照相同途径进行定制,但是它的标题、信息和新闻资料等的收集都是关于开发的。在此用户选择的信息决定了进入MSDN在线主页将看到的内容,就像在Microsoft主页中那样。

有两种途径可以进入定制页面:进入MSDN在线主页(msdn.microsoft.com)并点击页面顶部的Customize按钮;或者直接将浏览器指向msdn.microsoft.com/msdn-online/start /custom。无论使用何种方法进入,该页面如图3-5所示。

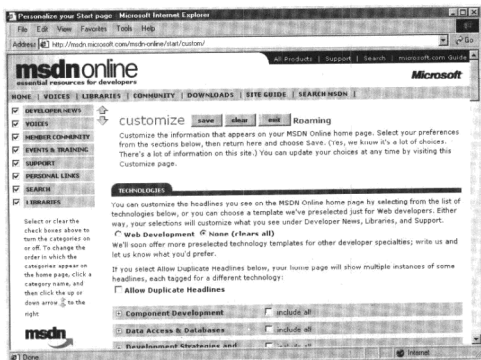


图3-5 MSDN在线的定制页面

如图3-5所示,该主页中有大量可选的技术资料。如果对Web开发感兴趣,可以选择在TECHNOLOGIES区域中靠近顶部的Web Development选项,从而选择面向Web技术的预定义子集。关于更多面向Win32库的技术,可以选择适当的技术选项。如果希望选择给定技术组的所有技术,可在该技术阴影标题区域中选中Include All复选框。

用户还可以从信息中选择包含哪些目录(由MSDN在线列出),以及它们的排列顺序。可用的目录包括:

Developer News

Support

Voices

Member Community

Events & Training

Personal Links

Search

Libraries

一旦用户定义了自己的配置文件，即定制了希望看到的MSDN在线内容，那么在用户每次进入MSDN在线主页时，MSDN在线将显示与用户配置文件相关的最新信息，包括按照指定顺序排列的已选目录。注意清除一个给定复选框（比如Libraries）将从用户MSDN在线主页中（并且不包括该目录的标题）清除该目录，但是不会从MSDN在线站点导航工具条中删除该目录。换句话说，如果用户清除某个目录，它将不会成为该用户定制的MSDN在线页面标题的一部分，但是它仍然是站点的一项功能。

最后，如果用户希望自己的配置文件一直可用，而无论使用的是哪台计算机，则可以要求MSDN在线创建漫游配置文件（roaming profile）。为MSDN在线创建漫游配置文件，可使该文件存储在MSDN在线的服务器上，和Windows 2000的漫游相似，这就使得该文件一直可用而无论使用的是哪台计算机。创建漫游配置文件的选项，可在定制MSDN在线主页时选择（该选项在任何时间都可用）。但是，创建漫游配置文件需要用户是MSDN在线的注册用户。关于如何成为MSDN在线的注册用户的更多信息，将在下面的“MSDN在线注册用户”小节中提供。

1. 浏览MSDN在线

一旦用户已经完成MSDN在线主页的定制，将得到最希望看到的标题，浏览MSDN在线很简单。在MSDN在线图标下的横幅，将起到导航工具条的作用，通过所带有的下拉菜单可访问MSDN在线的各个领域，如图3-6所示。

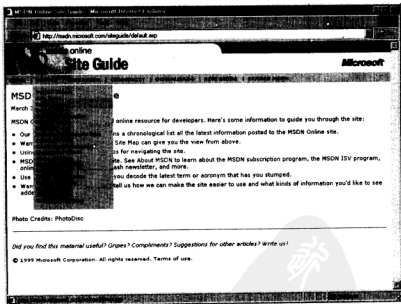


图3-6 MSDN在线中带有下拉菜单的导航工具条

可用的菜单目录（按照MSDN在线中可用的站点和功能分组）包括：

Home	Voices
Libraries	Community
Downloads	Site Guide
Search MSDN	

因为导航工具条将一直可用，而无论用户身处MSDN在线的何处，因此可以使用这一熟悉的菜单来浏览站点，仅需一次点击即可进入MSDN在线的任意领域。这些菜单目录对MSDN在线所提供的功能，按其作用和逻辑关系进行了分组。

2. MSDN在线的功能

MSDN在线中七个功能目录中都包括不同的站点，这些站点中包含了开发人员在访问MSDN在线时可以使用到的功能。

Home已经很熟悉了；在导航工具条中点击Home，将返回用户已定制（也许）的MSDN在线主页，向用户显示被其指定为感兴趣、希望阅读的最新技术标题。

Voices是构成MSDN在线杂志部分的专栏和论文的集合，可以通过msdn.microsoft.com/voices直接链接访问。Voices的主页如图3-7所示。

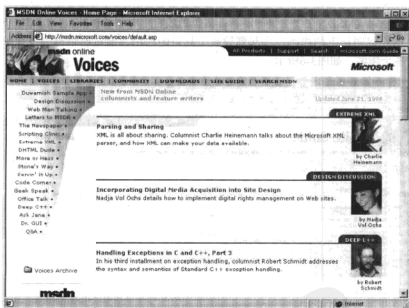


图3-7 Voices主页

在Voices站点中的每个“voice”，都对开发人员所面临的问题有自己的专题讨论。在Voices站点中，应用程序和Web开发人员都可以从一定规模的不同论文清单（并且经常更新）中，得到类似于杂志的论文。

Libraries是MSDN在线上参考资料所在的地方。Libraries站点分为两个部分：Library和Web

Workshop。划分参考资料的标尺是在MSDN还是在Site Builder Network中，也即，划分了是Windows应用程序开发还是Web开发。从Libraries菜单中选择Library，将按照传统的MSDN风格浏览页面，并对传统的MSDN参考资料提供访问。Library主页可以通过msdn.microsoft.com/library直接链接访问。选择Web Workshop，将进入稍微不同的Web Workshop浏览，如图3-8所示。Web Workshop主页可以通过msdn.microsoft.com/workshop直接链接访问。

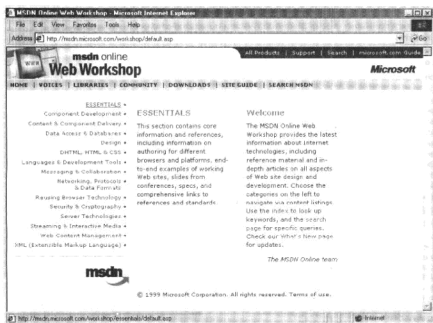


图3-8 Web Workshop主页，带有开始点的浏览清单

Community是为Windows和Web开发人员提供在线论坛的场所。在此可以共享开发思想和技术，可以找到或提出建议（通过MHM，即Members Helping Members），并且从在线特别兴趣小组（Online Special Interest Group, OSIG）中可以找到论坛发表观点，或者与其他开发人员交谈。Community站点包含各种有用素材，包括有特色的书籍、宣传资料和下载资料、实例学习等等。Community主页可以通过msdn.microsoft.com/community直接链接访问。图3-9提供Community主页的概貌。

Downloads站点让开发人员可以找到各种可用的适合下载的内容，比如工具、示例、图像和声音。Downloads站点还是MSDN订阅人员通过Internet得到最新发布内容的地方，用以更新其订购版内容。Downloads主页可以通过msdn.microsoft.com/downloads直接链接访问。Downloads主页如图3-10所示。

Site Guide恰如其名称的含义一样，是MSDN在线站点的向导，目的是帮助开发人员找到感兴趣的内容，并且包括对MSDN在线其他页面的链接，比如最新发布的文件列表、站点地图、词汇表以及其他有用链接。Site Guide主页可以通过msdn.microsoft.com/siteguide直接链接访问。

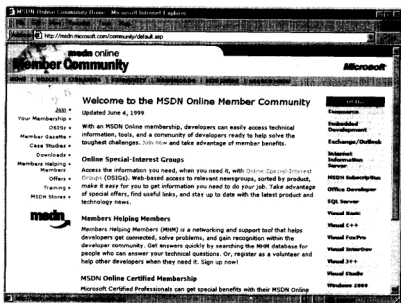


图3-9 Community主页

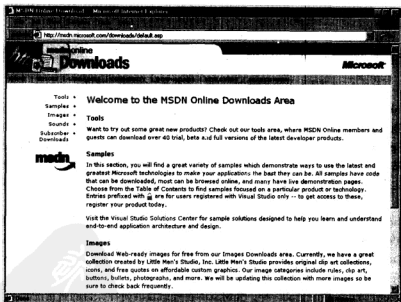


图3-10 Downloads主页

MSDN在线上的Search MSDN站点已经对先前版本做出改进,包括将搜索限制在资料库中(Library或者Web Workshop)的能力,以及其他性能调整能力。Search MSDN主页可以通过msdn.microsoft.com/search直接链接访问。Search MSDN主页如图3-11所示。

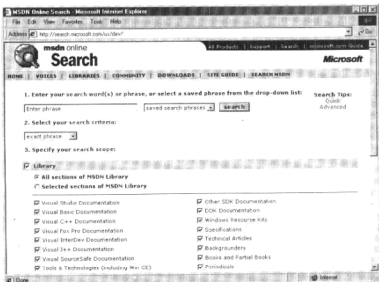


图3-11 Search MSDN主页

3. MSDN在线注册用户

读者可能已经注意到在访问MSDN在线的一些功能时(比如创建漫游的配置文件,这些配置文件用于某些组功能的使用许可证),要求用户是注册用户。和MSDN订购版不同,成为MSDN在线的注册用户并不需要任何耗费,仅需几分钟的注册时间。

MSDN在线的某些功能,要求在使用其服务之前进行注册。例如,若想成为OSIG的成员,就需要注册。这种功能本身就有足够的理由要求注册;和试图传呼开发伙伴来解决某个问题(尤其是发现他正好休假两天,而问题的期限是几个小时)不同,用户可以来到MSDN在线的Community站点,通过OSIG搜索,在点击之间即可得到答案。

成为一个注册用户有一系列的好处,比如可以选择在邮箱接收新闻快报。用户还可以得到各种类型的其他定期信息,比如讨论提醒,让用户知道何时专家将在MSDN在线Community站点有个专题。用户还可以基于不同的OSIG成员标记出新闻快报。笔者强烈建议各位读者也成为MSDN在线的用户,笔者就是一位注册用户,并且得到了大量的资源。

3.2 联系信息

微软向开发人员提供了最新的、简明的和有针对性的资料,使得开发人员能够尽可能高效地完成工作。除了提供Microsoft技术的参考资料之外,还提供来之不易的深入分析。

现在计划包括如下几套书籍：

- “Win32开发人员参考库”（Win32 Developer's Reference Library）
- “活动目录开发人员服务库”（Active Directory Services Library）
- “网络连接服务开发人员参考库”（Networking Services Library）
- “COM+开发人员参考库”（COM + Developer's Reference Library）

将来还会有哪些呢？计划中的专题，比如安全（Security）系列、语言参考（Language Reference）系列、MFC系列、BackOffice系列或者使用Microsoft产品的开发人员所需的其他相关专题，都有望奉献给读者。如果用户希望对本套书提供反馈意见，请发送E-mail到如下地址：

winprs@microsoft.com

如果发送的E-mail是关于某一套书籍的，请确保在主题栏中标注该书名称。例如，本套书的E-mail应该将主题标注为“Win32 Library”。不能保证每封信都会有回音，但是编者会阅读所有E-mail，并尽力确保用户的评论、看法或者（特别是）意见会被正确处理。

第4章 查找开发人员资源

对于Windows应用程序开发人员来说,存在大量的各种资源,它们能够对开发人员每天面对的大多数问题提供答案,但是搜索这些资源有时会比问题本身更为困难。本章针对如何尽可能多地向读者提供一步到位的可用的开发人员资源,从而使应用程序的开发工作变得更为简单。

Microsoft在通过MSDN和MSDN在线提供大量资料的同时,还通过出版系列丛书提供大量有侧重点的参考资料和开发提示,以及更多的可用信息。其中一些来自Microsoft,一些来自开发团体,还有一些来自从事专业开发服务的公司。在本章中读者将发现有哪些可选的开发资源,从而对可用资源更为熟悉。

Microsoft通过许多不同的媒体、渠道和途径,来提供开发人员资源。Microsoft资源提供的可扩展性反映了不同的环境需要。例如,在寻求编程项目中特定开发问题的解答时,用户可能不会去查参考手册,而会使用Microsoft其他资源之一。

4.1 开发商支持

Microsoft的支持站点覆盖了支持问题的很广范围,包括Microsoft的所有产品,但是这些站点中的大部分并不针对开发人员。不过,还是有一些站点是为支持开发人员而设计的;开发人员的Product Services Support页面,就是开发人员所需的搜索支持信息的集中地带。图4-1显示了开发人员的Product Services Support页面,可在www.microsoft.com/support/customer/develop.htm中看到。

注意对于Microsoft的支持有一系列的选择,包括从简单地在知识库(Knowledge Base)中在线搜索已知漏洞,到从Microsoft咨询服务(Microsoft Consulting Services)得到一手的咨询服务,以及介于它们之间的一切服务。图4-1中显示的Web页面是一个较好的开始点,可以从中得到关于Microsoft支持服务的更多信息。

来自Microsoft的Premier Support(首要支持)为开发人员提供广泛的支持,并且针对不同的Microsoft客户提供不同的支持包。Microsoft提供的首要支持包有:

- Premier Support for Enterprises
- Premier Support for Developers
- Premier Support for Microsoft Certified Solution Providers
- Premier Support for OEMs

如果读者是位开发人员,可能会需要这些内容。

来自Microsoft的Priority Annual Support(优先年度支持)针对经常需要向Microsoft询问支持问题的开发人员或公司,并且他们的问题需要被优先处理。Microsoft提供的优先年度支持包有三种:

- Priority Comprehensive Support

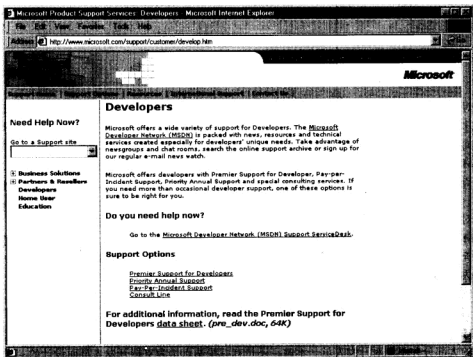


图4-1 开发人员的Product Support Services页面

- Priority Developer Support
- Priority Desktop Support

作为一位开发人员，最好的支持选择是Priority Developer Support。

注意，作为MSDN Professional订购版的一部分，Microsoft提供两次优先技术支持；作为MSDN Universal订购版的一部分，将提供有四次优先技术支持。

用户也可以通过Microsoft的支持Web站点向Microsoft工程师提交问题，但是如果问题有时间限制的话，请慎重考虑这一途径。或者可以考虑去MSDN在线，进入Community站点寻找关于开发问题的帮助。为了在线向Microsoft工程师提交问题，请访问support.microsoft.com/support/webresponse.asp。

4.2 在线资源

Microsoft还通过MSDN在线与开发人员的交流，提供扩展的开发人员支持。在MSDN在线的Community站点，用户可以看到以在线的形式覆盖各种问题的OSIG。为了进入MSDN在线的Community站点，请访问msdn.microsoft.com/community。

Microsoft的MSDN在线还提供知识库在线，作为Microsoft公司站点的个人支持中心（Personal Support Center）。可以在support.microsoft.com/support/search上搜索在线知识库。

Microsoft提供大量新闻组 (Newsgroups), 开发人员可以使用它了解关于新闻组特有的专题, 为创建Windows应用程序搜索信息提供另一种开发人员资源。要找到可用的新闻组, 以及与之取得联系, 请访问support.microsoft.com/support/news。

有好些新闻组可能是“Win32开发人员参考库”的读者所感兴趣的, 比如下列新闻组:

Microsoft.public.win32.programmer.*

Microsoft.public.vc.*

Microsoft.public.vb.*

Microsoft.public.platformsdk.*

Microsoft.public.cert.*

Microsoft.public.certification.*

当然, Microsoft并不是关于Windows开发技术相关的唯一的新闻组提供者。Usenet有许多各种类型的新闻组讨论在Windows平台上开发应用程序的主题。读者可以在访问任何其他新闻组时, 访问到关于Windows开发的新闻组; 一般情况下, 用户需要与其ISP联系以得到邮件服务器的名称, 然后使用新闻阅读应用程序以访问、阅读或者发送信息到Usenet新闻组。

4.3 学习产品

Microsoft提供大量产品, 以帮助开发人员为达到目标 (或者完成任务) 学习所需的特定任务或工具。比较适合开发人员的一条产品线被称做“精通系列”, 其产品面向广泛的开发主题, 提供完备的、组织良好的、交互式的教学工具。

Microsoft的“精通系列”包含由一组书籍和CD组成的交互式工具, 让读者掌握存在疑问的专题。如果想得到关于“精通系列”产品的更多信息, 或者希望了解“精通系列”所提供的内容, 请访问msdn.microsoft.com/mastering。

另外一些学习产品由其他商家提供; 比如其他出版商, 创建漫游类型的内容和应用程序的其他应用程序提供者, 以及针对特定技术发布音像制品 (包括磁带和Internet上的广播) 的公司。

关于使用某种语言 (比如Visual C++、Visual FoxPro或者Visual Basic)、使用某种操作系统, 或者对使用某种产品 (比如SQL Server或Commerce Server) 开发进行学习的另一条途径, 是阅读已经被授权为Microsoft认证方案开发商 (Microsoft Certified Solution Developer, MCSD) 的资料。若没有足够的准备时间或对此没有兴趣, 就不必勉强要通过认证。换句话说, 没必要为了证明这些资料有用而去获得证书; 实际上它们不会讲什么新东西, 有时甚至不如读者了解得多。事实上这些用以通过认证的教程, 是严格的和高难的, 并且太过于面向细节了。如果希望获得证书, 必须很好地掌握应用程序编程的基础和关于Windows平台中面向开发人员的信息。

要获得MCSD证书, 开发人员需要经过一系列核心考试, 然后从众多可选的选修考试中选择一个专题, 以完成证书的要求。核心考试可从一组考试中选取; 必须通过总共三门才能完成核心考试的要求。有考生通常选择的课程, 比如Visual C++开发或是Visual Basic开发。核心考试及其考试号如下:

00083880

桌面应用程序开发 (选一):

- 使用Microsoft Visual C++ 6.0设计和实现桌面应用程序 (70-016)。
- 使用Microsoft Visual FoxPro 6.0设计和实现桌面应用程序 (70-155)。
- 使用Microsoft Visual Basic 6.0设计和实现桌面应用程序 (70-176)。

分布式应用程序开发 (选一):

- 使用Microsoft Visual C++ 6.0设计和实现分布式应用程序 (70-015)。
- 使用Microsoft Visual FoxPro 6.0设计和实现分布式应用程序 (70-156)。
- 使用Microsoft Visual Basic 6.0设计和实现分布式应用程序 (70-175)。

解决方案体系:

- 需求分析和定义解决方案体系 (70-100)。

选修考试是候选人从大量附加考试中所选取的, 只有通过才能完成MCSD考试的要求。下面列出可选的MCSD选修考试。

可选的选修考试有:

- 未被使用作为核心考试的任何桌面或者分布式考试。
- 使用Microsoft SQL Server 7.0和Microsoft Decision Support Services 1.0设计和实现数据仓库。
- 使用Microsoft Foundation Class Library 4.0库开发C++应用程序。
- 在Microsoft Foundation Class Library 4.0应用程序中实现OLE。
- 在Microsoft SQL Server 6.5上实现数据库设计。
- 使用Microsoft SQL Server 7.0设计和实现数据库。
- 使用Microsoft FrontPage 98设计和实现Web站点。
- 使用Microsoft Site Server 3.0商业版设计和实现商业解决方案。
- Windows 95下的Microsoft Access和Microsoft Access Developer's Toolkit。
- 使用Microsoft Office 2000和Microsoft Visual Basic for Application设计和实现解决方案。
- 使用Microsoft Access 2000设计和实现数据库应用程序。
- 使用Microsoft Outlook 2000和Microsoft Exchange Server 5.5设计和实现协作解决方案。
- 使用Microsoft Visual InterDev 6.0设计和实现Web解决方案。
- 使用Microsoft Visual FoxPro 6.0设计和实现分布式应用程序。
- 使用Microsoft Visual FoxPro 6.0设计和实现桌面应用程序。
- 使用Microsoft Visual Basic 5.0开发应用程序。
- 使用Microsoft Visual Basic 6.0设计和实现分布式应用程序。
- 使用Microsoft Visual Basic 6.0设计和实现桌面应用程序。

关于这些考试的最好消息, 并不是有诸多的考试可供选择, 而是由于有许多认证需要通过考试, 因此有大量的书籍和其他资料, 介绍如何达到通过考试所需的知识水平。

掌握这些信息的途径是, 为这些考试中的一门或多门而学习, 但不要认为书籍越厚越好。应学习那些准备考试的材料, 这些准备考试的材料来自于各种类型的出版社, 包括Microsoft出版社、IDG、Sybex等等。大多数备考课本中还有模拟考题, 供读者自己检测资料的掌握程度。

读者可能会对自己掌握的程度之低而惊讶,即使有些人可能已经在这一领域的复杂项目中工作了相当长的时间也会如此。

当然,这些考试的要求以及考试本身,都会随着时间而变化;会有更多的选修考试,基于软件的修订版本的考试将被废止等等。要想得到关于认证过程的更多信息,或者关于考试的更多信息,请访问www.microsoft.com/train_cert/dev。

4.4 会议

和其他产业中一样,以Microsoft和开发团体作为发起人,一年到头都有针对不同专题的会议(遍布全世界范围)。可能没有人能够参加如此多的会议,但是每个会议通常针对一个特定专题;因此,围绕某个开发专题,开发人员可以选择一些符合工作需要和感兴趣的会议。

每年,MSDN本身主持或者发起将近一百场会议(其中一些是区域性的而在不同地点有重复,因此有可能一次会议多次举行)。其他会议在某个集中地点举行,比如较大的一次——专业开发人员会议(Professional Developers Conference, PDC)。无论开发人员关心哪个会议,Microsoft都有一个集中的站点来提供会议信息。如果希望了解在感兴趣的开发领域中有哪些会议或者哪些事件正在发生,请访问events.microsoft.com。

4.5 其他资源

对于Windows应用程序开发人员来说,还有其他可用资源,其中一些可能是某些开发人员的主要依靠资源,而对其他开发人员而言却闻所未闻。本章所描述的关于开发人员资源信息已经远远超出了“开发人员资源简介”这一初衷,但是,虽然如此,笔者相信在这里还有某些未曾涉及的内容。

毫无疑问,还有许多其他资源。如果读者拥有资源并希望与其他人员共享,请按如下地址向我们发送E-mail,可能其他人会从中受益:

winprs@Microsoft.com

如果发送的E-mail与某一特定有用资源有关,请在主题栏中键入“Resources”。不能保证每封信都会有回音,但是笔者会阅读信中的所有内容,并尽力确保读者的资源信息得以考虑。

第5章 Win32编程的常见错误

本章介绍常见编程错误，可以帮助程序员避开简单的编程陷阱。本套书的每一卷的第5章都包含这些内容，每一卷的内容不同：

第1卷：概述及解决方案总结

第2卷：避免无效验证

第3卷：RPC错误及内核模式的分类符

第4卷：缓冲区溢出及其他错误

第5卷：内存滥用及计算错误

并非所有的这些陷阱都会出现在Win32编程过程中（例如，有些是基于网络服务的）。然而，既然在所有Windows应用程序中都应该避免上述常见编码错误，那么在此提供这些错误的清单可以让用户在使用本套书时更加得心应手。

本书是“Win32开发人员参考库”套书的第4卷，本章中所提供的错误说明与例子将有助于读者在从事开发工作的过程中避免出现缓冲器溢出与其他错误。下面将对这些错误情况进行说明。

5.1 缓冲器溢出

缓冲器溢出可能会导致各种各样的问题，开发人员的简单错误或者他人对系统进行的攻击都有可能会导致出现缓冲器溢出问题。虽然避免缓冲器溢出的工作并不困难，但如果在软件开发过程中没有考虑避免缓冲器溢出问题，则有可能导致非常严重的后果。为了有效地避免缓冲器溢出问题，应该在软件开发的过程中遵守以下设计规则：

- 在访问某个缓冲器时，应该记住检查实际的缓冲器大小，而不能只是查看关于缓冲器的已知最大大小。
- 注意算术计算中的溢出问题，从而确保缓冲器检查时不会因为算术计算的溢出而出错。
- 验证对枚举型数进行算术计算的结果值仍然是有效枚举数。
- 不能期望缓冲器大小已经经过了测试，而应该进行实际的测试工作。

为了避免缓冲器溢出，还必须注意以下问题：

- 在使用偏移地址时，应该确保结果地址位置没有越过缓冲器的两个边界。
- 在进行复杂的大小计算时，应该确保缓冲器总大小大于固定头尺寸。
- 注意没有以NULL结尾的字符串，对于这种字符串，如果已知字符串大小，就应该使用这个信息。
- 在完成枚举数计算之后，检查计算结果是否在枚举数的最小值到最大值之间。
- 在进行外部数据与内部数据的比较时，首先进行数据尺寸的比较，然后使用其中尺寸较小的进行比较。

5.1.1 简单缓冲器溢出

简单缓冲器溢出问题的最佳解决方案就是在对缓冲器引用之前首先检查缓冲器的边界。但是，有两种情况应该引起特别的重视。

一种需要引起注意的情况是在向堆栈缓冲器中写入数据。如果在向堆栈缓冲器中写入数据的过程中超过了缓冲器的边界，就可能会使得返回地址指向一个任意值，从而导致错误代码的执行。另一种需要引起注意的情况是无终止字符串。在许多情况下（例如内核模式与网络结构），字符串都是直接用大小指定，而不是给出NULL作为终止符。在这些情况下，就不能利用NULL终止符，而必须使用字符串大小。

一条通用的规则就是，应该记住根据缓冲器的上界与下界检查缓冲器的访问，而不能仅仅根据已知的最小/最大值进行检查。

实例

```
NTSTATUS
AppBug(IN UNICODE_STRING *String)
{
    NTSTATUS Status;
    UNICODE_STRING CapturedString;
    WCHAR *Buffer = NULL;

    try {
        ProbeForRead(String,
                      sizeof (UNICODE_STRING),
                      4);
        CapturedString = *String;
        ProbeForRead(CapturedString.Buffer,
                      CapturedString.Length,
                      sizeof (WCHAR));
        Buffer = ExAllocatePoolWithQuotaTag(PagedPool,
                                           CapturedString.Length,
                                           'guB');

        RtlCopyMemory(Buffer,
                      CapturedString.Buffer,
                      CapturedString.Length);
        Status = InternalFoo(Buffer);
        ExFreePool(Buffer);
    } except (EXCEPTION_EXECUTE_HANDLER) {
        if (Buffer) {
            ExFreePool(Buffer);
        }

        Status = GetExceptionCode();
    }
    return Status;
}
```

说明

在这个例子中，向内部函数传递了一个字符串缓冲器，但在传递过程中，既没有给出字符串大小，也没有使字符串以0字符结尾。

5.1.2 大小上溢或下溢

在许多实例中，特别是在网络代码中，所传递的缓冲器通常是这种形式：缓冲器头大小固定，但尾部大小可变。对于这种类型的缓冲器，常常需要进行复杂的大小计算，并且应该经过仔细的合法性检查。这种缓冲器最常见的出错情况就是，为缓冲器的可变长度部分指定了一个很大（几乎为负）的大小时，就可能会使缓冲器头部分与尾部分的大小之和小于缓冲器大小。在进行合法性检查成功后，就会导致巨大的内存部分要复制到更小的缓冲器中。另一个常见出错情况就是发送比头尺寸还小的部分报文，只要对缓冲器大小检查工作稍做修整，就能够纠正这种错误。

实例

```
typedef struct _INFO {
    ULONG FlagBits;
    ULONG DataSize;
    UCHAR Data [1 ];
}INFO,*PINFO;

NTSTATUS
AppGetInfo(
    PINFO Info,
    ULONG Len
)
{
    if (Len < (ULONG)FIELD_OFFSET(INFO,Data)){
        return STATUS_INFO_LENGTH_MISMATCH;
    }
    if (Len < (ULONG)FIELD_OFFSET(INFO,Data)+Info->DataSize){
        return STATUS_INFO_LENGTH_MISMATCH;
    }
    [-]
}
```

说明

上面代码实例的问题在于第二个if语句中的加法可能会溢出，而溢出可能会使得即使缓冲器大小不是以保存足够的数据，但对缓冲器大小的测试仍然会成功。只要重新安排上面的if语句，就能够使得缓冲器检查工作能够正确地进行：

```
if (Len - (ULONG)FIELD_OFFSET (INFO, Data)<Info->DataSize) {
    return STATUS_INFO_LENGTH_MISMATCH;
};
```

说明

由于在前面的代码中第一个语句已经进行了测试，因此在上述代码中可以在进行第一个减法时不进行下溢检查。

5.1.3 枚举类型的误用

枚举类型的值都是有界的，这就意味着某些操作（特别是加法与减法操作）可能会产生导致非法内存引用的值。因此在进行任何算术操作之后，应该对枚举类型的最大和最小值进行检查。

实例

```
typedef enum {
    FirstEntry,
    SecondEntry
}INFOCLASS;

INFOCLASS InformationClass;
[...
Length = InfoLengths [InformationClass - 1];
```

说明

考虑 `InformationClass == FirstEntry`，这个表达式的计算值为0。

5.1.4 使用内部长度进行外部输入的比较

某些应用程序组件为组件所使用的数据结构维护有一个内部长度值。这一点对于应用程序组件内部的数据是很好的；但是，当把应用程序的这种内部长度应用于外部输入数据时，就可能会出现问題。如果数据长度可以不同，就应该使用内部长度与外部长度中较小的一个。如果长度必须相同，那么当出现长度大小不一致时，参数将被拒绝。

实例

```
case IOCTL_CRASH_SYSTEM:
    //
    //IOCTL_CRASH_SYSTEM is METHOD_BUFFERED,so the
    //buffer we get from the I/O system is the max of
    //the input and output buffer sizes.
    //
    Buffer =pIrp ->AssociatedIrp.SystemBuffer;
    RtlCopyMemory(Buffer,
        GlobalSource,
        sizeof GlobalSource);
    pIrp ->IoStatus.Information =sizeof GlobalSource;
```

说明

如果这个代码实例中的 `RtlCopyMemory()` 函数调用没有因为写入过多的数据而导致系统崩溃，那么I/O系统代码将这些数据拷贝到用户模式中时，则可能会导致系统崩溃。Outputlength可能会为0。

5.2 其他错误

本节讨论那些在任何应用程序代码中都可能比较常见，但又不便于归纳到某个错误类型中的错误。下面列出了开发人员在开发应用程序的过程中应该注意的其他问题：

- 在将一种类型的数据传递给另一种数据类型时应谨慎。
- 对于复杂表达式中出现的运算优先权问题应仔细检查。
- 确保组合条件中所有部分相等价（每个结果应该执行相同的代码），或者在适当条件下进行了特殊处理。
- 检查所有指针参数是否为NULL（特别是可选参数）。
- 不要在代码中对字符串进行硬编码（例如Administrators）。
- 应该对常变数据进行多次检查。
- 在需要顺序操作时，请求使用上锁机制。
- 注意（并消除或减少）与通用接口（例如，GetLastError与返回句柄的函数）不一致的情况。

5.2.1 数据类型转换的危险

如果没有进行仔细的检查，就将某个输入值分配给另一数据类型，就有可能导致许多问题。例如，假设指针为4字节长时，如果将应用程序移植到64位操作系统中，就可能会导致出现严重的问题。而如果将某输入数据指定为浮点类型，则可能会导致最严重的问题，这是因为许多位组合形式在浮点中并不是合法的，因此输入数据可能不能成为合法的浮点数。最后，将某输入数据分配给其它数据类型可能会产生不同的行为结果，因为数据类型的符号位设置取决于数据类型是否为带符号数；对于无符号数，使用的是0扩展策略；对于带符号数，使用的是符号位扩展策略。

一般来说，在应用程序中进行数据类型转换时应该尽可能少地做假设。将一个指针转换为长整型数据时可能会导致数值的截取，从而使得产生看似正确实际错误的数据。从用户模式中传递而来的浮点数应该假设其包含有各种可能的位模式，而不能只定为浮点数。

实例

```
NTSTATUS BufferCheck(
    BUFFER InputBuffer,
    USHORT InputBufferLength
)
{
    USHORT AddressLength;
    if (InputBufferLength > (USHORT)(InputBuffer->DataOffset +
        InputBuffer->DataLength)){
        //
        //The data buffer checks out OK. Get the
        //length of the target address (first
        //USHORT after the data buffer).
        //
```

```
AddressLength =*(USHORT *)((PCHAR)InputBuffer +
    InputBuffer->DataOffset +InputBuffer->DataLength));
```

[...]

说明

在这个例子中，DataOffset与DataLength是ULONG类型的数据，这两个变量的和被转化为USHORT类型的数据，从而执行与USHORT类型数据InputBufferLength的比较。由于对值进行了截取，因此在执行条件判断时可能会成功，但在AddressLength获取到相应值时，却可能出现对缓冲器引用越界的情况，因为所讨论的变量没有经过再次转换。

5.2.2 操作符运算优先级

许多常见问题都是由于开发人员对操作符运算优先级理解不当所造成的，特别常见的就是对&与!以及==与!=理解不正确。等号具有更高的运算优先级，因此，代码if (a & c1 == c2)表示的是if (a & (c1 == c2))。为了避免出现这种情况，在必要时应使用括号，或者必须严格检查表达式的运算优先级是否正确。

实例

```
if (Value & CONSTANT == CONSTANT) {
    RetVal = ERROR_INVALID_PARAMETER;
}
```

说明

这样，上面例子中的条件句实际上就是if (Value & !0)。有一个名为TYPO的PERL脚本能对此种错误情况进行检查。一个比较智能的编译器应该能够对下面的语句进行优化，一个智能化程度更高的编译器则应该能给出关于这种问题的警告信息。

5.2.3 条件终止混乱

使用复合条件后，如果后续的代码中假设了此条件中的某部分成立，则会导致条件终止混乱。这种特定的编程错误在带有复合终止语句的while循环或for循环中经常出现。

实例

```
while (index < BufferLength && Buffer [index] != '\0') {
    index++;
}
StringLength = strlen (Buffer);
```

[...]

说明

这个例子中的循环似乎在检查缓冲器，确保缓冲器不会越界的情况下由NULL结尾，但是，后面一条语句紧接着假设缓冲器的终止标记已经找到，这样就使得第二个条件满足while的循环终止条件。但是，如果第一个条件短语满足了条件终止需求，那么strlen调用将可能会产生超过缓冲器长度的返回值。

5.2.4 OPTIONAL参数的误用

OPTIONAL参数可能为NULL,但是,某些函数在验证此参数不是NULL的情况下就取消引用OPTIONAL参数;或者,在某些路径上检查此参数是否为NULL,而在另一些路径上又不进行这种检查。避免这种常见编程错误的方法很简单,就是在使用OPTIONAL参数之前,检查此参数是否为NULL。

实例

BOOL

```
DoubleDup(
    char *StringSrcA,
    char *StringSrcB,
    char *StringDstA OPTIONAL,
    DWORD *pLenA,
    char *StringDstB OPTIONAL,
    DWORD *pLenB)
{
    if ((StringSrcA == NULL) || (StringSrcB == NULL)) {
        return FALSE;
    }

    if (StringDstA == NULL) {
        *pLenA = strlen(StringSrcA) + 1;
        *pLenB = strlen(StringSrcB) + 1;

    } else {
        strncpy(StringDstA,
                StringSrcA,
                *pLenA);
        strncpy(StringDstB,
                StringSrcB,
                *pLenB);
    }

    return TRUE;
}
```

说明

请读者考虑StringDstA!=NULL与StringDstB==NULL二个条件短语。

5.2.5 返回值混乱与不一致

在Win32 API中包含有几个函数,虽然设计者最初希望这些函数能够对所有系统API函数通用,但事实并非如此。其中两个最容易被误用的就是INVALID_HANDLE_VALUE与GetLastError。在大多数Win32 API函数之后,都可以调用GetLastError,但有些函数(例如,注册表API函数)之后则不能调用SetLastError。同样,Nt* API函数也不能调用SetLastError。

INVALID_HANDLE_VALUE只能从Win32文件系统API函数(CreateFile、FindFirstFile等等)返回值。如果将INVALID_HANDLE_VALUE传递给GetKernelSecurity()函数,则返回当前进程的安全性参数值,这是因为在这种情况下INVALID_HANDLE_VALUE==GetCurrentProcess()。

为了避免出现这些常见的编程错误,必须仔细检查适当的错误返回码。有些Win32处理函数返回NULL,而有些则返回INVALID_HANDLE_VALUE。GetLastError()函数所导致的不一致问题必须被修正,但应该确保使用返回码进行错误检查,而不仅仅是使用GetLastError()函数进行错误检查。

实例

```
HANDLE
OpenFile(
    ...
)
{
    NtOpenFile(_);
    if (GetLastError() == STATUS_SUCCESS){
        //
        //Open succeeded. Now read the data.
        //
    }
    ...
}
```

说明

对NtOpenFile()函数的调用之后并没有调用SetLastError()函数,因此,对GetLastError()函数的调用将导致返回上一个调用了SetLastError()函数的返回值。如果上一次对SetLastError()的调用成功,那么就会导致错误的响应动作。

5.2.6 不应该轻易使用易变对象

在任何多线程环境下,如果对全局数据进行多次检查并且试图得到相同的结果,就有可能导致同步问题。如果某个内核或网络服务器需要根据对某个易变对象的多次检查来作出决策,那么必须确保对象的不同值不会导致算法的失效。避免这个问题的最佳方法就是避免执行多次相同的查询。如果需要进行多次查询,那么应该确保查询所得到的不同结果不会导致出错。例如,如果对某个文件的访问当前被判定为是能够进行的,那么不应该据此判断以后对这个同名文件访问仍然会成功。

实例

```
NTSTATUS
ReadWriteFileByName(
    PCHAR FileName,
    BUFFER ReadBuffer,
    BUFFER WriteBuffer
)
{
    HANDLE FileHandle;
```

```

//
//Check that the calling user has
//sufficient privileges.
//
if (!UserHasReadWritePrivileges(FileName)){
    return(STATUS_ACCESS_DENIED);
}
FileHandle =OpenForRead(FileName);
[Code to read data from the file into the supplied buffer ...]
CloseHandle(FileHandle);
if (WriteBuffer !=NULL){
    //
    //A write buffer was supplied; reopen the
    //file and write the indicated data.
    //
    FileHandle =OpenForWrite(FileName);
}
[...]
```

结果

在执行读打开操作与写打开操作之间的这个时间段内，例子中的文件使用特权可能会发生改变。由于这是一个特权组件，并且没有执行任何假冒操作，因此代码可能会试图向一个已经被标记为只读状态的文件写入数据。

5.2.7 避免旋转锁顺序错误

如果按照错误的顺序使用旋转锁（或者其他上锁/互斥机制）可能会创建最终导致计算机死锁的定时窗口。许多组件在IRP取消例程中都有可能会存在这种死锁的可能性，因为在IRP例程中可能会没有去除隐含请求的CancelSpinlock，而导致对旋转锁的使用。为了避免出现这种情况，应该按照一个一致的顺序来请求使用旋转锁，即使是在有旋转锁被隐含请求的情况下也应该如此。

实例

```

VOID
LockingFunction1(PSESSION Session)
{
    PCONNECTION Connection;
    KIRQL OldIrql;
    KeAcquireSpinLock(&Session->Lock,&OldIrql);
    Connection =Session->Connection;
    if (Connection->Session !=Session){
        KeReleaseSpinLock(&Session->Lock,OldIrql);
        return;
    }
    KeAcquireSpinLockAtDpcLevel(&Connection->Lock);
    [...]
    KeReleaseSpinLockFromDpcLevel(&Connection->Lock);
    KeReleaseSpinLock(&Session->Lock,OldIrql);
}
```

```

    return;
}
VOID
LockingFunction2(PCONNECTION Connection)
{
    PSESSION Session;
    KIRQL OldIrql;
    KeAcquireSpinLock(&Connection->Lock, &OldIrql);
    Session = Connection->Session;
    if (Session == NULL){
        KeReleaseSpinLock(&Connection->Lock, OldIrql);
        return;
    }
    KeAcquireSpinLockAtDpcLevel(&Session->Lock);
[-]

```

说明

如果LockingFunction1()与LockingFunction2()分别在同时请求进行会话与连接上锁，那么，两个线程都会死锁地等待另一个线程释放自己在下一步需要的锁。

5.2.8 判定管理组成员关系

许多应用程序在允许用户执行某个操作之前，首先要判断这个用户是否是一个管理员，但在执行判断成员关系的过程中可能会出现错误。在Administrators组中判定成员关系的最常见方法就是创建一个适当的SID，并且在用户令牌中检查这个SID。但是，对于严格的令牌来说，这种检查是不够的。另一种常见的方法就是指定名称“Administrators”来查找SID，但这种方法也不能进行精确的定位，因此不是最佳的方法。最佳方法就是使用CheckTokenMembership()来检查用户在任何组中的成员关系。

实例

```

//
//See if this user is an Administrator.
//
BOOL IsMember = FALSE;
TCHAR SidBuffer [128 ],RefDom [128 ];
ULONG AdminSidSize =128 *sizeof (TCHAR);
ULONG RefDomSize =128 *sizeof (TCHAR);
ULONG Size =0;
NTSTATUS Status;
PSID AdminSid =(PSID)SidBuffer;
PSID_NAME_USE NameUse;

if (LookupAccountName(NULL,
    TEXT("Administrator"),
    &AdminSid,
    &AdminSidSize,
    &RefDom,

```

```

        &RefDomSize,
        &NameUse)){
    if (!GetTokenInformation(Token,
        TokenGroups,
        NULL,
        0,
        &Size)){
        Groups =RtlAllocateHeap(RtlProcessHeap(),
            0,
            Size);
        if (Groups){
            if (GetTokenInformation(Token,
                TokenGroups,
                Groups,
                Size,
                &Size)){
                for (i =0;i <Groups->GroupCount;i++){
                    if (RtlEqualSid(AdminSid,
                        Groups->Groups [i].Sid){
                        *IsAdmin =TRUE;
                        break;
                    }
                }
            }
            RtlFreeHeap(RtlProcessHeap(),
                0,
                Groups);
        }
    }
}

```

如果需要修复在这个代码实例中出现的问题，应该将上面的代码写成以下形式：

```

BOOL IsMember;
PSID AdminSid;
PSID_IDENTIFIER_AUTHORITY NtAuthority =SECURITY_NT_AUTHORITY;

if (AllocateAndInitializeSid(&NtAuthority,
    2,
    SECURITY_BUILTIN_DOMAIN_RID,
    DOMAIN_ALIAS_RID_ADMINS,
    0,0,0,0,0,0,
    &AdminSid)){
    if (!CheckTokenMembership(Token,
        AdminSid,
        &IsMember)){
        IsMember =FALSE;
    }
    GlobalFree(AdminSid);
}

```

5.3 解决方案小结

对上面出现的这些常见编程错误，应当有一致的解决方案，下面将小结如何避免本章中所描述的那些问题。

避免缓冲器溢出问题的方法：

- 1) 简单缓冲器溢出：在访问一个缓冲器时，应该记住检查缓冲器的实际大小，而不是使用某些已知的缓冲器最大值。
- 2) 缓冲器大小上溢或下溢：在使用偏移地址时，应该确保实际的地址位置不会超出缓冲器的两个边界。
- 3) 误用枚举类型：对于复杂的大小计算操作，应该确保总大小应大于固定头尺寸。
- 4) 使用内部长度进行外部输入比较：注意出现那种没有带NULL终止符的字符串。如果已经有了关于字符串的大小信息，应该使用这个信息。

避免其他错误的方法：

- 1) 类型转换的危险：在将输入数据转换成另一种类型时，应该十分谨慎。
- 2) 操作符运算优先级：对于复杂的表达式，应该仔细检查操作符运算优先级问题。
- 3) 条件终止混乱：应该确保组合条件中所有的条件语句都等价（每个结果应该执行相同的代码），或者在适当的情况下进行了特殊的考虑。
- 4) OPTIONAL 参数的误用：检查所有指针参数是否为NULL（特别是可选参数）。
- 5) 返回值混乱和不一致：不要在代码中对字符串进行硬编码（例如Administrators）。
- 6) 不要轻易使用易变对象：应该注意到对易变数据进行多次检查时的数据改变问题。
- 7) 避免旋转锁顺序问题：总是按照一致的顺序请求上锁。
- 8) 判定管理组成员关系：注意（同时应该努力消除或减少）与通用接口（例如，GetLastError与返回句柄的函数）的不一致性问题。



第二部分 通用控件参考

几乎在所有Microsoft产品中，以及在许多由第三方软件开发商所提供的应用程序产品中，通用控件都是用户界面的重要部分。但是，由于存在大量的控件，使得查找自己所需的信息用来执行某个常见或重要任务并不是一件简单的事情。下面将描述在软件开发过程中应该考虑的版本控制问题，同时还说明其他几个对于开发人员特别重要的信息。

获取关于列表视图、工具条以及树状视图控件的信息

为了压缩本书内容，在本卷中没有讨论以下三个通用控件：列表视图控件、工具条控件与树视图控件。可以说，关于这三个控件的介绍就可能会有本书那样厚。为了能够向读者提供更加完全而且有用的参考资料系列，编者以更为压缩的形式提供了关于这三个控件的介绍信息。本套书第1卷中的随书光盘包含有这三个控件的介绍（同时，在这个光盘中还有卷中所有其他控件以及许多参考信息的介绍）。

通用控件概述

通用控件（common control）就是一系列由通用控件库所实现的窗口，其中，通用控件库是包含在Microsoft Windows操作系统中的一个动态链接库（DLL）。与其他控件窗口一样，通用控件也是应用程序用来与其他窗口实现I/O任务的子窗口。

使用通用控件

大多数通用控件都属于通用控件DLL中所定义的一个窗口类。这个窗口类以及相应的窗口过程定义了控件的属性、外观、行为。为了确保通用控件DLL被加载，必须在应用程序中使用InitCommonControlsEx函数。如果需要创建一个通用控件，那么就应该在调用CreateWindowEx函数时指定窗口类名，或者在对话框模板中指定适当的类名。

1. DLL版本

所有32位版本的Windows都包含有一个名为Comctl32.dll的通用控件DLL。自从第一次被引入以来，这个DLL已经经历了多次升级。这个DLL的每个后续版本都能够支持先前版本中所具有的功能与应用程序编程接口（API）。但是，每个新的DLL版本中都包含有许多新特性以及更为庞大的API。因此，应用程序必须能够知道在系统中安装的是哪个版本的Comctl32.dll，从而确保使用只有这个相应DLL版本能够支持的功能与API。

由于新版本的通用控件是与Internet Explorer一同发布的，从而使得当前的Comctl32.dll版本与随同操作系统一同发布的Comctl32.dll版本不一致。实际上，系统中所使用的Comctl32.dll版本可能是更新的版本。这样，应用程序就不能仅仅知道自己所运行的操作系统版本，同时还必须

明确地知道系统中Comctl32.dll的版本。关于通用控件版本以及如何判断系统中所安装Comctl32.dll版本的详细信息,请参见下面的“Shell与通用控件版本”小节。

不同通用控件版本的数据结构大小

通用控件在功能上的不断加强,导致需要对许多数据结构进行扩充。这样,由于数据结构的扩充,导致不同版本的Commctrl.h之间的某些数据结构大小也不一样。由于大多数通用控件数据结构都使用数据结构大小作为自己的一个参数,因此,如果不能对数据结构的大小进行正确的识别,那么就可能会导致消息调用的失败或函数调用的失败。为了解决这个问题,系统中定义了数据结构大小常数这一概念,从而确定不同Comctl32.dll的版本。下面的列表中定义了新的数据结构大小常量:

控 件	常 量
HDITEM_V1_SIZE	4.00版本中HDITEM数据结构的大小
LVCOLUMN_V1_SIZE	4.00版本中LVCOLUMN数据结构的大小
LVHITTESTINFO_V1_SIZE	4.00版本中LVHITTESTINFO数据结构的大小
LVITEM_V1_SIZE	4.00版本中LVITEM数据结构的大小
NMLVCUSTOMDRAW_V3_SIZE	4.70版本中NMLVCUSTOMDRAW数据结构的大小
NMTTDISPINFO_V1_SIZE	4.00版本中NMTTDISPINFO数据结构的大小
NMTVCUSTOMDRAW_V3_SIZE	4.70版本中NMTVCUSTOMDRAW数据结构的大小
PROPSHEETHEADER_V1_SIZE	4.00版本中PROPSHEETHEADER数据结构的大小
PROPSHEETPAGE_V1_SIZE	4.00版本中PROPSHEETPAGE数据结构的大小
REBARBANDINFO_V3_SIZE	4.70版本中REBARBANDINFO数据结构的大小
TTTOOLINFO_V1_SIZE	4.00版本中TOOLINFO数据结构的大小
TVINSERTSTRUCT_V1_SIZE	4.00版本中TVINSERTSTRUCT数据结构的大小

2. 通用控件样式

每种类型的通用控件都有一系列控件样式(或称为风格),利用这些控件样式能改变控件的外观与行为。通用控件库中也包含有一系列可以作用于两个或多个类型通用控件的控件样式。

3. 通用控件消息

由于通用控件是窗口,因此应用程序可以使用消息(例如,WM_GETFONT或WM_SETTEXT)对这些窗口进行处理。此外,每个通用控件的窗口类中也支持一系列控件所特有的消息,应用程序能够使用这些消息来对控件进行处理,例如,可以利用消息发送函数将消息传递给控件。此外,某些通用控件还有一系列宏集合,应用程序可以使用这些宏而不是调用消息发送函数来完成消息发送的动作。一般来说,宏比函数更容易使用。

当系统中的颜色设置发生了改变时,Windows就将向所有的顶级窗口发送WM_SYSCOLORCHANGE消息,然后,顶级窗口必须将WM_SYSCOLORCHANGE消息转发给通用控件;否则,控件就不能接收到颜色改变的信息。这样,就能够确保通用控件所使用的颜色能够与其他用户界面对象的颜色保持一致。例如,一个工具条控件使用3D Object颜色绘制自己的按钮。如果用户改变了3D Object颜色,但这个WM_SYSCOLORCHANGE消息没有被转发给这个工具条,那么该工具条按钮将维持自己原有的颜色(甚至有可能改变成为原有颜色与新颜色的组合形式),而系统中其他按钮的颜色却已经改变。

4. 通用控件通告消息

通用控件是一个子窗口，当在控件中发生了某个事件（例如，来自用户的输入）时，它能够向父窗口发送通告消息。应用程序利用这些通告消息来判断用户希望执行的动作。除了轨迹条控件是使用WM_HSCROLL消息与WM_VSCROLL消息向自己的父窗口告知改变信息之外，其他通用控件都是以WM_NOTIFY消息的形式向父窗口发送通告消息的。其中，WM_NOTIFY消息中的IPParam参数是NMHDR数据结构的地址，或者是以NMHDR作为其第一个数据成员的更大数据结构的地址。在NMHDR数据结构中包含有通告消息代码，并且在这个数据结构中给出了标识发送此通告消息的通用控件信息，数据结构中的其他数据成员（若存在）则根据通告消息代码的不同而不同。

通用控件通告消息能够支持ANSI格式与UNICODE格式，系统将通过发送WM_NOTIFYFORMAT消息方法来判断所使用的是哪种通告消息格式。如果需要确定一个通告消息格式，那么对于ANSI通告消息，返回的是NFR_ANSI；对于Unicode通告消息，返回的是NFR_UNICODE。如果没有处理这个消息，那么系统将调用IsWindowUnicode函数来判断所使用的通告消息格式。由于Windows 95与Windows 98对这个函数调用总是返回FALSE，因此这两个系统在缺省情况下使用ANSI通告消息。

注意 并非所有的控件都发送WM_NOTIFY消息。实际上，标准Windows控件（编辑控件、组合框、列表框、按钮、滚动条、静态控件）都不发送WM_NOTIFY消息。如果需要判断某个控件是否能够发送WM_NOTIFY消息，或者判断如果能够发送这个消息，应该使用哪个通告消息代码，请参见相应控件的文档说明。

每种类型的通用控件都有一个相应的通告消息代码集合，同时，在通用控件库中也提供了可以被多种通用控件所使用的通告消息代码。如果需要了解某个通告消息能够使用哪些通告消息代码，以及所使用的通告消息格式，请参见相应的通用控件文档说明。

Internet Explorer中升级的通用控件

Internet Explorer中的通用控件能够支持以下新特性：

1) 初始化通用控件。

现在，能够使用InitCommonControlsEx函数初始化通用控件，这个函数能够指定为应用程序初始化哪个控件，而不必对所有的控件进行初始化。虽然系统中还提供了对InitCommonControls函数的支持，但在新的应用程序中应该使用InitCommonControlsEx函数。

2) 新的通用控件样式。

定义了4种新的通用控件样式，分别是：CCS_LEFT、CCS_RIGHT、CCS_VERT和CCS_NOMOVEX。

Shell与通用控件版本

本节说明如何判定应用程序正在运行的Shell或通用控件DLL版本，以及如何控制应用程序使用指定的Shell或通用控件DLL版本。

DLL版本号

Shell与通用控件文档中所描述的编程元素几乎都包含在三个DLL中：Comctl32.dll、

Shell32.dll、Shlwapi.dll。由于通用控件的功能不断加强，从而使得在不同的DLL版本中具有不同的功能。版本号用来表示编程元素首次在所指定的DLL版本中引入，并且能够在所有的后续DLL版本中使用。如果没有指定版本号，就表示编程元素在所有的版本中可用。下表列出了不同的DLL版本，以及发布情况：

版 本	DLL	发 布 平 台
4.00	全部	Microsoft Windows 95/Windows NT 4.0
4.70	全部	Microsoft Internet Explorer 3.x
4.71	全部	Microsoft Internet Explorer 4.0 (请参见说明2)
4.72	全部	Microsoft Internet Explorer 4.01和Windows 98 (请参见说明2)
5.00	Shlwapi.dll	Microsoft Internet Explorer 5 (请参见说明3)
5.00	Shell32.dll	Microsoft Windows 2000 (请参见说明3)
5.80	Comctl32.dll	Microsoft Internet Explorer 5 (请参见说明3)
5.81	Comctl32.dll	Microsoft Windows 2000 (请参见说明3)

说明1：4.00版本的Shell32.dll与Comctl32.dll在Windows 95与Windows NT 4的原始版本中已经存在，新版本的Commctl.dll则随着所有的Internet Explorer一同发布。Shlwapi.dll在Internet Explorer 4.0中首次发行，因此其第一个版本就是4.71版本。在Internet Explorer 3.0版本中没有对Shell进行更新，因此，Shell32.dll不存在4.70版本。虽然4.71版本以及4.72版本的Shell32.dll已经在相应的Internet Explorer版本中发行，但系统中未必安装了这些版本（请参见说明2）。对于后续版本，三个DLL的版本号并不完全相同。一般来说，应该假设这三个DLL可能会有不同的版本号，然后分别对其进行测试。

说明2：所有安装了Internet Explorer 4.0或4.01的系统都应该有相应的Comctl32.dll与Shlwapi.dll（分别是4.71版本或4.72版本）。但是，对于Windows 98之前的系统，Internet Explorer 4.0与4.01中可能安装了集成Shell，也可能没有安装集成Shell。如果没有安装集成Shell，那么Shell32.dll就相应地没有得到更新。也就是说，4.71版本或4.72版本的Comctl32.dll与Shlwapi.dll在系统中如果存在并不能保证相应版本的Shell32.dll在系统中也存在。所有的Windows 98系统都具有Shell32.dll的4.72版本。

说明3：5.80版本的Comctl32.dll以及5.0版本的Shlwapi.dll随同Internet Explorer 5一同发布。除了Windows 2000之外，只要安装有Internet Explorer 5的系统中，就已经安装了5.80版本的Comctl32.dll以及5.0版本的Shlwapi.dll。Internet Explorer 5并没有更新Shell，因此在Windows NT、Windows 95或Windows 98系统中都不存在5.0版本的Shell.dll。将在Windows 2000中发行5.0版本的Shell32.dll、5.0版本的Shlwapi.dll以及5.81版本的Comctl32.dll。

使用DllGetVersion确定版本号

自从4.71版本开始，Shell与通用控件DLL就提供DllGetVersion函数支持。应用程序可以调用这个函数，来确定系统中DLL的版本号，这个函数将返回一个包含有版本信息的数据结构。

注意 DLL未必一定提供了DllGetVersion函数；因此，在使用之前应该对其进行测试。

对于Windows 2000以前的系统，DllGetVersion返回一个DLLVERSIONINFO数据结构，在这个数据结构中包含有主版本号、从版本号、创建编号以及平台ID。对于Windows 2000及其更新

系统, DllGetVersion则返回一个DLLVERSIONINFO2数据结构。在这个数据结构中, 包含有用来自定义服务包的QFE数字, 并且提供了比DLLVERSIONINFO更灵活的版本号比较功能。由于DLLVERSIONINFO2数据结构的第一个数据成员就是一个DLLVERSIONINFO2数据结构, 因此新的数据结构能够向后兼容。

使用DllGetVersion

下面的函数实例将加载一个指定的DLL, 并且试图调用其中的DllGetVersion函数。如果调用成功, 就使用一个宏将DLLVERSIONINFO2数据结构中的主版本号与从版本号合成到一个DWORD数据中, 然后将这个DWORD返回给调用函数。如果这个DLL没有提供DllGetVersion函数支持, 那么函数将返回0。对于Windows 2000及其更新系统, 可以对这个函数进行修改, 从而使得能够处理由DllGetVersion所返回的DLLVERSIONINFO2数据结构。这时, 就应该使用包含在ullVersion数据成员中的信息与版本号、创建编号以及服务包发行版本进行比较。MAKEDLLVERULL宏可以简化将这些值与包含在ullVersion中的值进行比较的过程。

```
#define PACKVERSION(major,minor)MAKELONG(minor,major)

DWORD GetDllVersion(LPCTSTR lpszDllName)
{
    HINSTANCE hinstDll;
    DWORD dwVersion =0;

    hinstDll =LoadLibrary(lpszDllName);
    if(hinstDll)
    {
        DLLGETVERSIONPROC pDllGetVersion;

        pDllGetVersion =(DLLGETVERSIONPROC)
        GetProcAddress(hinstDll,"DllGetVersion");

        /*Because some DLLs may not implement this function,you
        *must test for it explicitly.Dependent on the particular
        *DLL,the lack of a DllGetVersion function may
        *be a useful indicator of the version.
        */

        if(pDllGetVersion)
        {
            DLLVERSIONINFO dvi;
            HRESULT hr;

            ZeroMemory(&dvi,sizeof(dvi));
            dvi.cbSize =sizeof(dvi);
            hr =(*pDllGetVersion)(&dvi);

            if(SUCCEEDED(hr))
            {
                dwVersion =PACKVERSION(dvi.dwMajorVersion,
                dvi.dwMinorVersion);
            }
        }
    }
}
```

```

        FreeLibrary(hinstDll);
    }
    return dwVersion;
}

```

下面的代码段演示了如何使用GetDllVersion来测试Comctl32.dll是否是4.71版本或更新版本。

```

if(GetDllVersion(TEXT("comctl32.dll")) >= PACKVERSION(4,71))
{
    //Proceed
}
else
{
    //Use an alternate approach for older DLL versions
}

```

工程版本

为了确保应用程序能够与目标版本的comctl32.dll以及shell32.dll相互兼容，应该在头文件中添加一个版本宏，用来定义、排除或重新定义不同版本的DLL。这个宏的名称是_WIN32_IE，开发人员必须负责对这个宏进行定义，并且必须定义成为一个十六进制数。下面列出了当前可用的版本号以及每个版本号对应用程序的影响：

版本号	描述
0x0200	应用程序能够与Comctl32.dll以及shell32.dll 4.00版本及其后续版本相兼容。同时，应用程序不能实现Comctl32.dll 4.00版本之后所添加的新功能
0x0300	应用程序能够与Comctl32.dll以及shell32.dll 4.70版本及其后续版本相兼容。同时，应用程序不能实现Comctl32.dll 4.70版本之后所添加的新功能
0x0400	应用程序能够与Comctl32.dll以及shell32.dll 4.71版本及其后续版本相兼容。同时，应用程序不能实现Comctl32.dll 4.71版本之后所添加的新功能
0x0401	应用程序能够与Comctl32.dll以及shell32.dll 4.72版本及其后续版本相兼容。同时，应用程序不能实现Comctl32.dll 4.72版本之后所添加的新功能
0x0500	应用程序能够与Comctl32.dll 5.80版本及其后续版本以及shell32.dll与Shlwapi.dll 5.0版本及其后续版本相兼容。同时，应用程序不能实现Comctl32.dll 5.80版本之后以及Shell32.dll与Shlwapi.dll 5.0版本之后所添加的新功能
0x0501	应用程序能够与Comctl32.dll 5.81版本及其后续版本以及shell32.dll与Shlwapi.dll 5.0版本及其后续版本相兼容。同时，应用程序不能实现Comctl32.dll 5.81版本之后以及Shell32.dll与Shlwapi.dll 5.0版本之后所添加的新功能

如果在工程中没有定义这个宏，那么宏将会自动地定义成为0x0500。如果需要定义一个不同的值，应该将下面的代码添加到make文件的编译器指令中（当然，需要用正确的版本号来替换下一行中的0x400）：

```
/D_WIN32_IE=0x0400
```

另一种定义这个宏的方法就是在源代码中添加如下所示的一行，并且将这一行加入到shell与通用控件头文件之前（当然，需要用正确的版本号来替换下一行中的0x400）。例如：

```

#define _WIN32_IE 0x0400
#include <commctrl.h>

```

第6章 使用通用控件

6.1 创建可定制工具条

为了给用户访问各种工具的方便性，大多数Microsoft Windows应用程序都使用工具条控件，但是，静态工具条存在一些缺点，例如，窗口中空间有限不足以显示所有可用工具。

关于这个问题的解决方案就是使应用程序中的工具条成为可定制的工具条。这样，用户就可以根据自己特有的方便性，按照实际需要在工具条中移动、添加、删除工具。

如果需要打开工具条的可定制特性，就必须在创建工具条控件时使工具条具有CCS_ADJUSTABLE通用控件样式。有两种基本方法，可以用来进行工具条的定制：

- 定制对话框。这种由系统所提供的对话框是最简单的定制方法，向用户提供一个图形用户界面，用来进行图标添加、删除或移动。
- 拖放工具。如果实现了工具条的拖放特性，就能够允许用户在工具条上将某个工具从一个位置移动到另一个位置，或者，将某个工具拖动到工具条之外，从而删除这个工具。这种方法虽然提供了对工具条进行快速组织的途径，但不能向工具条中添加工具。

根据应用程序的实际需要，可以实现其中一种或两种工具条定制方法。

这两种工具条定制方法都不能提供一种内置的机制（例如，Cancel或Undo按钮），将工具条恢复到先前的状态。因此，必须明确地使用工具条控件API来存储工具条在定制之前的状态，这样，如果在后来需要恢复工具条的初始状态，就可以使用所存储的信息。

下面将讨论如何打开定制对话框与拖放工具的工具条定制功能，同时还简单地讨论了工具条状态的保存与恢复问题。

6.1.1 定制对话框

工具条控件中所提供的定制对话框可以为用户给出一个简单的方法，向工具条中添加、移动或删除工具。用户只要双击工具条，就能够激活定制对话框。如果应用程序需要激活定制对话框，则必须向工具条控件发送一条TB_CUSTOMIZE消息。图6-1给出了工具条定制对话框的示例。

其中，右边列表框中所列出的工具是那些当前在工具条中存在的工具，初始情况下，这个列表框中将列出创建工具条时在工具条中所包含的工具。左边的列表框中包含有可以添加到工具条中的工具，应用程序负责对这个列表框进行更新以及记录工具条中当前存在的工具。

实现定制对话框

工具条控件将向其父窗口发送一条TBN_BEGINADJUST通告消息，从而告知应用程序，工具条控件将要激活定制对话框。然后，再发送一条TBN_INITCUSTOMIZE通告消息。如果不希望工具条显示Help按钮，那么就必须处理这条通告消息并且返回TBNRF_HIDEHELP。

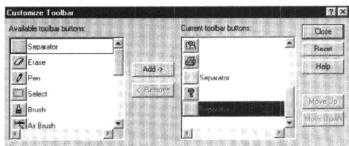


图6-1 工具条定制对话框

然后，工具条控件就将按顺序发送以下通告消息序列，从而收集初始化对话框所需的信息：

1) 向工具条上的每个按钮发送一条TBN_QUERYINSERT通告消息，从而判定应该将按钮插入到何处。如果返回的是FALSE，那么将阻止此按钮插入到通告消息中所指定按钮的左边。如果所有的TBN_QUERYINSERT通告消息都返回FALSE，那么对话框将不会被显示出来。

2) 向工具条中的每个工具发送一条TBN_QUERYDELETE通告消息，如果返回TRUE，那么就表示工具可以被删除；否则，如果返回FALSE，那么就表示工具不能被删除。如果所有的工具能够被删除，那么就不必处理这条通告消息。

3) 发送一系列TBN_GETBUTTONINFO通告消息，从而获取可用工具的列表。如果需要向列表中添加一个工具，就填充通告消息所传递的NMTOOLBAR数据结构并且返回TRUE。如果没有任何工具可以添加，则返回FALSE。

然后，对话框就被显示出来，从而可以开始进行工具条的定制工作。

对话框被显示之后，根据用户所执行动作的不同，应用程序可能会接收到许多不同的通告消息：

- TBN_QUERYINSERT。每次用户在工具条上修改工具的位置，或者添加一个工具时，应用程序都将接收到这条通告消息。如果返回FALSE，则表示不允许在那个位置上插入此工具。
 - TBN_DELETINGBUTTON。每次用户将要从工具条中删除某个工具时，应用程序将接收到这条通告消息。
 - TBN_CUSTHELP。用户单击了Help按钮之后，应用程序将接收到这条通告消息。
 - TBN_TOOLBARCHANGE。用户添加、移动或删除某个工具之后，应用程序将接收到这条通告消息。
 - TBN_RESET。用户单击Reset按钮之后，应用程序将接收到这条通告消息。
- 对话框被析构之后，应用程序将接收到一条TBN_ENDADJUST通告消息。

6.1.2 拖放工具

用户也可以在按住SHIFT键的同时，在工具条上将一个按钮拖动到另一个位置，从而对工具

条中的按钮进行重新布局。拖放工作由工具条控件自动处理。在拖动按钮的过程中，按钮将显示为一个幻像，在放下按钮之后，工具条将进行重新布局。用户不能用这种方法添加按钮，但能够将某个按钮拖动到工具条之外，从而删除整个按钮。

虽然工具条控件能够自动地完成上述操作，但仍然会向应用程序发送两条通告消息：TBN_QUERYDELETE与TBN_QUERYINSERT。为了实现对拖放过程的控制，必须按照以下方法处理这两条通告消息：

- 只要用户企图移动按钮，在按钮的幻像被显示之前，应该发送TBN_QUERYDELETE通告消息。如果返回FALSE，则表示阻止按钮的移动；如果返回TRUE，那么用户就能够移动这个工具，或者将工具拖动到工具条之外，从而删除这个工具。在允许用户移动某个工具之后，就无法阻止用户删除此工具。但是，如果用户删除某工具，工具条控件将向应用程序发送TBN_DELETINGBUTTON通告消息。
- 当用户试图在工具条上放下某个按钮时，将发送TBN_QUERYINSERT通告消息。如果需要阻止这个按钮被放置在通告消息所指定按钮的左边，就应该返回FALSE。如果用户将工具拖放到工具条之外，将不发送这条通告消息。

如果用户没有按下SHIFT键，就试图拖动某个按钮，那么工具条控件将不处理拖放操作。但是，仍然会向应用程序发送一条TBN_BEGINDRAG通告消息，表示拖动操作的开始；发送TBN_ENDDRAG通告消息，表示拖动操作的结束。如果希望打开这种形式的拖放功能，那么应用程序就必须对这些通告消息进行处理，提供必要的用户界面，并且修改工具条从而反映用户所执行的修改操作。

6.1.3 保存与恢复工具条状态

在定制工具条之后，可能会希望将工具条返回到其先前状态。但是，在用户定制工具条时，工具条控件不会自动地记录自己的定制前状态。因此，为了便于以后进行工具条的恢复，应用程序必须能够保存工具条状态。其方法如下：

如果需要保存一个工具条状态，那么就应该向工具条控件发送一条iParam参数被设置为TRUE的TB_SAVERESTORE消息。缺省情况下，工具条控件将自动地保存自己的信息。对于通用控件版本5.80及其后续版本，只要为TBN_SAVE通告消息实现一个处理程序，就能够获取对保存操作的更多控制。

如果需要恢复一个工具条状态，那么就应该向工具条控件发送一条iParam参数被设置为FALSE的TB_SAVERESTORE消息。缺省情况下，通用控件将向应用程序发送一系列TBN_GETBUTTONINFO通告消息，从而请求获取所保存的每个按钮信息。对于通用控件版本5.0及其后续版本，只要为TBN_RESTORE通告消息实现一个处理程序，就能够获取对恢复操作的更多控制。

6.2 创建同位工具提示

文本字符串通常被用来进行标注小对象。但是，如果需要显示的有用信息文本字符串过长，

那么这些文本字符串就有可能超过对象显示区域的边界,从而导致文本字符串被剪辑。一个常见的例子就是Microsoft Windows Explorer中的文件名,如图6-2所示。

如果这些标注被剪辑,那么就会严重地限制其有用性。但是,在图6-2所给出的例子中,如果用户将鼠标停留在某个文件名上,就能够见到相应的完全文件名。因为在用户执行这个动作时,将会在被剪辑文件名之上出现带有完全文件名的同位工具提示(in-place toolTip),如图6-3所示。



图6-2 被剪辑文件名的例子



图6-3 显示有完全文件名的同位工具提示

普通工具提示与同位工具提示之间的区别在于工具提示窗口的定位方法不同。缺省情况下,当鼠标停留在与工具提示相关联的某个区域上时,工具提示将显示在区域的旁边。但是,由于工具提示也是窗口,因此只要调用SetWindowPosition函数,就能够将其定位到任何位置上。创建同位工具提示,就是要将工具提示窗口定位到能够覆盖相应文本字符串的位置即可。

定位同位工具提示

在定位同位工具提示时,需要知道三个矩形:

- 完全包围标注文本的矩形。
- 包围工具提示文本的矩形。工具提示文本的内容与完全标注文本的内容相同,并且在通常情况下具有相同的大小与字体。因此,这两个文本矩形的大小相同。
- 工具提示窗口的矩形。这个矩形通常比工具提示文本矩形稍大。

图6-4中显示了这三个矩形的示意图,其中,标注文本的被隐藏部分用灰色背景表示。

为了创建同位工具提示,必须对工具提示文本矩形进行定位,从而使其能够覆盖标注文本矩形。进行这两个矩形的对齐操作比较简单:

- 1) 定义标注文本矩形。
- 2) 定位工具提示窗口,从而使得工具提示文本矩形能够覆盖标注文本矩形。

在实际使用中,通常只要将两个文本矩形的左上角进行对齐即可。如果试图将工具提示文本矩形的大小进行修改,从而使得能够与标注文本矩形大小相匹配,有时可能会导致工具提示显示出现问题。

这种方法所存在的一个问题就是不能直接定位工具提示文本矩形。相反，必须定位工具提示窗口矩形，使其左上角比标注文本矩形的左上角更加靠上而且更加靠左，从而使得两个文本矩形能够重叠。也就是说，必须能够知道工具提示窗口矩形与其中所包含文本矩形之间的偏移量。目前，还没有一种简单方法能够确定这种偏移量。

使用TTM_ADJUSTRECT定位工具提示

在通用控件的5.0版本中添加了一条新消息TTM_ADJUSTRECT，从而简化了同位工具提示的使用。只要在发送消息的同时给出工具提示将要覆盖的标注文本矩形的坐标，消息就能够返回进行工具提示窗口矩形定位之后的坐标。

下面的代码段中演示了如何在TTN_SHOW处理程序中使用TTM_ADJUSTRECT，从而显示同位工具提示。应用程序将私有变量fMyStringIsTruncated设置为TRUE，从而表示标注文本被截取。处理程序将调用一个由应用程序所定义的函数GetMyItemRect，获取标注文本的矩形。这个矩形是利用TTM_ADJUSTRECT消息传递给工具提示控件的，然后，此消息将返回相应的窗口矩形。最后，调用SetWindowPosition来定位覆盖在标注文本之上的工具提示。

```
case TTN_SHOW:

    if (fMyStringIsTruncated){
        RECT rc;

        GetMyItemRect(&rc);
        SendMessage(hwndToolTip, TTM_ADJUSTRECT, TRUE,
            (LPARAM)&rc);
        SetWindowPos(hwndToolTip,
            NULL,
            rc.left, rc.top,
            0, 0,
            SWP_NOSIZE | SWP_NOZORDER |
            SWP_NOACTIVATE);
    }
```

可以看出，上面的例子并没有改变工具提示的大小，而只是改变了其位置。两个文本矩形将在左上角对齐，但并不一定具有相同的大小。实际上，两个矩形之间的差别通常是很小的，因此对于大多数应用场合，笔者推荐使用这种定位方法。当然，从原则上说可以使用SetWindowPos修改工具提示的大小与位置，但这样做可能会导致不可预测的后果。

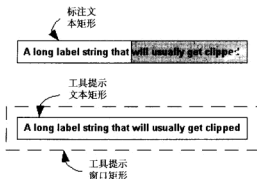


图6-4 定位同位工具提示时所使用的三个矩形

6.3 创建Internet Explorer样式的工具条

Microsoft Internet Explorer中的一个关键用户界面特性就是工具条。这个工具条不仅能够让用户使用许多功能，而且还允许用户定制工具条的布局，从而适应自己的个人偏爱。图6-5显示了Internet Explorer工具条，并且说明了其中的某些关键特性。

这个工具条实质上包含有由四个区段所组成的Rebar控件：三个工具条与一个菜单条。由于这个工具条是利用通用控件API实现的，因此开发人员能够创建带有任意功能的工具条。在这个文档中将讨论Internet Explorer工具条的特性以及如何在应用程序中实现这个工具条。



图6-5 Internet Explorer工具条

6.3.1 Rebar控件

Internet Explorer工具条中的底层结构是由Rebar控件提供的，这个控件为用户提供了一种定制工具集合布局的方法。每个Rebar控件中都包含有一个或多个区段，每个区段一般都是长而窄的矩形，在这个矩形中包含有一个子窗口（通常是工具条控件）。

Rebar控件在一个矩形区域内显示区段，通常是在窗口的顶部显示。这个矩形被分隔成为一个或多个条，其高度与区段的高度相同。每个区段可以在一个单独的条上，也可以将多个区段放置在相同的条上。

Rebar控件为用户提供了两种对工具进行布局的方法：

- 每个区段在其左边都有一个夹钳（gripper），当两个或多个区段在同一个条上，并且超出了窗口的宽度范围时，就需要使用夹钳。只要对夹钳进行左右拖动，就能够控制为每个区

段所分配的空间大小。

- 利用拖放的操作方法，用户就能够在Rebar显示矩形内移动区段。然后，Rebar控件将修改自己的显示，从而适应新的区段布局方式。如果所有的区段都从某个条中被移出，那么Rebar控件的高度将减少，从而使得可视区域加大。
- 应用程序可以按照需要添加区段或删除区段。典型情况下，应用程序能够允许用户通过使用View菜单或上下文菜单来选择将要显示的区段。

如果一个条上的区段组合宽度超过了窗口的宽度，Rebar控件将按照实际需要对自己的宽度进行调整。有些工具可能会被邻接的区段所覆盖。

5.80版本的通用控件提供了一种方法，能够使得对另一个区段所覆盖的工具可以被用户所访问。如果在区段的REBARBANDINFO数据结构fStyle数据成员中设置了RBBS_USECHEVRON标志位，那么将会为那些被覆盖的工具条显示一个人字形按钮。当用户单击人字形按钮时，将显示一个菜单，允许他/她使用被隐藏的工具。图6-6是从Internet Explorer 5.0所截取的一个屏幕示例，在这个图中就显示了被地址条所覆盖的标准工具条菜单。

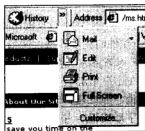


图6-6 当地址条覆盖了标准工具条时，所显示的部分菜单

由于每个区段中都包含有一个控件，因此可以通过控件API为控件提供更大的灵活性。例如，可以实现工具条定制，从而允许用户在工具条上进行按钮的添加、移动或删除。

1. 实现Rebar控件

Internet Explorer工具条中的大多数特性实际上都是在单个区段中实现的。Rebar控件自身的实现过程比较简单：

1) 利用CreateWindowEx创建Rebar控件，并且将dwExStyle设置为WS_EX_TOOLWINDOW，将lpClassName设置为REBARCLASSNAME。Internet Explorer可以使用下列窗口样式：

- CCS_NODIVIDER
- CCS_NOPARENTALIGN
- RBS_BANDBORDERS
- RBS_DBLCLKTOGGLE
- RBS_REGISTERDROP
- RBS_VARHEIGHT
- WS_BORDER
- WS_CHILD
- WS_CLIPCHILDREN
- WS_CLIPSIBLINGS
- WS_VISIBLE

同时，还必须按照应用程序的实际需要设置其他参数。

2) 利用CreateWindowEx或特定的控件创建函数（例如CreateTollbarEx）创建一个控件。

3) 填充REBARBANDINFO数据结构中的数据成员, 从而为控件初始化一个区段。在此过程中, 应该为fStyle数据成员设置RBBS_USECHEVRON样式, 从而确保打开了人字形按钮功能。

4) 利用RB_INSERTBAND消息, 将区段添加到Rebar控件中。

5) 为其余的区段重复执行步骤2~4。

6) 为Rebar通告消息实现处理程序。特别地, 应该对RBN_CHEVRONPUSHED通告消息进行处理, 从而能够在单击人字形按钮时显示下拉菜单。关于更多信息, 请参见下面“处理人字形按钮”部分。

缺省情况下, 区段中将包含有一个夹具。如果需要使得区段不使用夹具, 那么就必须在REBARBANDINFO数据结构的fStyle数据成员中设置RBBS_NOGRIPPER标志位。关于实现Rebar控件的更多信息, 请参见第22章。

2. 处理人字形按钮

在用户单击一个人字形按钮时, Rebar控件将向应用程序发送一条RBN_CHEVRONPUSHED通告消息。在随同通告消息一同传递的NMREBARCHEVRON数据结构中包含有区段标识符以及人字形按钮所占据的矩形的RECT数据结构。处理程序必须能够判定哪些按钮被隐藏并且在弹出菜单上显示相应的命令。

下面的过程列出了如何处理RBN_CHEVRONPUSHED通告消息:

1) 向Rebar控件发送一条RB_GETRECT消息, 从而为所选择的区段获取当前约束矩形。

2) 向区段的工具条控件发送一条TB_BUTTONCOUNT消息, 从而获取按钮数目信息。

3) 从最左端的按钮开始, 分别向工具条控件发送TB_GETITEMRECT消息, 从而获取按钮的约束矩形。

4) 将区段与按钮矩形传递给IntersectRect函数, 然后, 这个函数将返回一个与按钮的可视区域相对应的RECT数据结构。

5) 将按钮矩形以及按钮可视区域的矩形传递给EqualRect函数。

6) 如果EqualRect函数返回TRUE, 那么表示整个按钮可见。对工具条上的下一个按钮重复执行步骤3~5。如果EqualRect函数返回FALSE, 那么就表示这个按钮至少被部分地隐藏, 同时说明后面的所有按钮都被完全隐藏。然后继续下一步执行。

7) 为每个被隐藏的按钮创建弹出菜单。

8) 利用TrackPopupMenu显示弹出菜单。然后, 使用传递给RBN_CHEVRONPUSHED通告消息的人字形按钮矩形定位这个弹出菜单。一般来说, 弹出菜单应该出现在人字形按钮的正下方, 并且与其左边界对齐。

9) 处理菜单命令。

6.3.2 工具条

Internet Explorer工具条的复杂性主要表现在组成Rebar区段中控件的实现上, Internet Explorer通常能够显示四种区段:

- 菜单条。

- 标准工具条。
- 链接条。
- 地址条。

包括菜单条在内的所有这些区段，实际上都包含有工具条控件。本节将讨论标准工具条与链接工具条的实现。菜单条的实现比较复杂，将在6.4节中单独讨论。

本节将介绍为了增加控件的可使用性，在Internet Explorer中加入的新工具条功能。

1. 下拉按钮

下拉按钮能够支持多个命令，当用户单击下拉按钮时，按钮不是直接执行某个命令，而是弹出一个菜单，然后，用户就能够从这个菜单中选择某个命令执行。图6-7显示了Internet Explorer标准工具条中的下拉按钮与菜单。

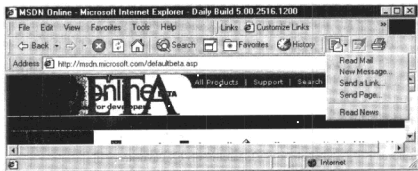


图6-7 Internet Explorer中标准工具条上的下拉按钮与菜单

只要在按钮的TBBUTTON数据结构fStyle数据成员中添加一个样式标志位，就能够为任何按钮添加下拉功能。有三种下拉按钮样式，这些样式都能够在Internet Explorer中使用：

- 普通的下拉按钮具有BTNS_DROPDOWN样式。这种按钮在外观上与普通按钮相同，但在用户单击这个按钮时，按钮将不是激活一个命令，而是显示一个菜单。
- 简单的下拉箭头按钮具有BTNS_WHOLEDROPDOWN样式，在这种样式下，按钮图像或文本的旁边将显示一个箭头。除了在外观上有这点不同之外，这种下拉按钮与普通下拉按钮相同。在上面的示例中所给出的Mail按钮实例就是一个简单下拉箭头按钮。
- 下拉箭头按钮在BTNS_DROPDOWN样式的基础上添加了TBSTYLE_EX_DRAWDDARROWS扩展样式，从而使得在按钮文本或图像的旁边有一个独立的箭头，这个按钮样式将简单下拉箭头按钮与标准按钮的功能结合起来。如果用户单击箭头，那么将显示菜单，然后用户就能够从菜单中所给出的几个命令中选择某个执行。如果用户直接单击按钮，那么将导致缺省的命令得到执行。图6-8显示了Internet Explorer中的Back按钮，这个按钮就是使用独立的箭头作为下拉按钮实现。

当用户单击普通下拉按钮或简单下拉箭头按钮时，工具控件将向应用程序发送一条TBN_DROPDOWN通告消息。应用程序接收到这条消息之后，在响应这条消息时，就创建



与显示菜单，并且处理所选择的命令。

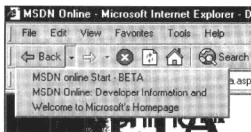


图6-8 Internet Explorer中的Back按钮



图6-9 标准按钮及文本

当用户单击下拉箭头按钮旁边的箭头时，工具控件将向应用程序发送一条TBN_DROPDOWN通告消息。同时，应用程序应该按照处理其他两种下拉按钮的方法一样来处理这条通告消息。如果用户单击主按钮，应用程序将接收到一条WM_COMMAND消息，在这条消息中包含有按钮的命令ID，这就如同处理标准的按钮过程一样。典型情况下，应用程序在响应这条消息时，将处理下拉菜单中的第一个命令。当然，也可以要求应用程序按照指定的方法进行消息响应。

2. 列表样式的按钮

对于标准按钮，如果向其中添加文本，那么文本将显示在位图的下面。如图6-9所示中显示Internet Explorer中带有标准按钮文本的Search按钮以及Favorites按钮。

在Internet Explorer 5中，使用了TBSTYLE_LIST样式，在这种样式下，文本显示在位图的右边，从该减少了按钮的高度，但同时增加了视图区域的宽度。下面演示了Internet Explorer 5中TBSTYLE_LIST样式的Search按钮以及Favorites按钮（见图6-10）。

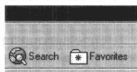


图6-10 文本显示在图右边

3. 人字形按钮

当用户在Rebar控件中重新布局区段时，可能会有部分工具条被覆盖。如果所创建的区段具有RBBS_USECHEVRON样式，那么Rebar控件将能够在工具条的右边显示一个人字形按钮。用户单击这个人字形按钮之后，就能够显示一个包含有被隐藏工具的菜单。关于如何实现人字形按钮的更多讨论，请参见6.3.1节中“处理人字形按钮”部分。

4. 热跟踪

如果打开了热跟踪功能，那么当鼠标移动到某个按钮之上时，这个按钮就变成“热按钮”。热按钮通常显示一个不同的图像，从而与其他按钮区分开来。缺省情况下，热按钮会被显示为在工具条上被提升（凸起）的形式。当一个新的按钮成为热按钮时，应用程序将接收到一条TBN_HOTITEMCHANGE通告消息。图6-11演示了Internet Explorer 5中的Search按钮以及Favorites按钮，其中Search按钮为热按钮。除了具有被提升的外观之外，按钮的灰色位图将会被一个其他颜色的位图所取代。

为了打开热跟踪功能，必须在创建工具条控件时使用TBSTYLE_FLAT样式或TBSTYLE_LIST样式。之所以称为是flat（平面）工具条，这是因为其中的按钮通常不会被高亮显示。工具条中的各个位图分别被邻接地显示出来，只有在热按钮状态下，才具有真正按钮那样的外观。这两个样式是透明的，也就是说，图标背景色就是所在客户窗口的颜色。

如果需要按钮处于热状态时显示不同的位图，那么必须为工具条中的所有按钮创建包含有热按钮图像的image list（图像列表），并且这些图像的大小与顺序应该与缺省图像列表相同。然后，为了设置热按钮图像列表，还必须向工具条控件发送一条TB_SETIMAGELIST消息。

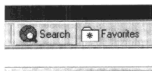


图6-11 热按钮

6.4 创建Internet Explorer样式的菜单条

初步一看，读者可能会发现Microsoft Internet Explorer 5.0中的菜单条与标准菜单没什么区别。但是，在使用上，却有很大的不同。图6-12显示了选中Tools菜单之后的Internet Explorer菜单条。

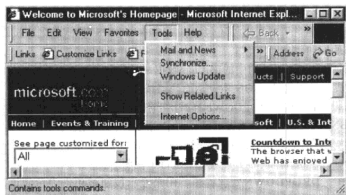


图6-12 Internet Explorer菜单条上所选中的Tools菜单

实际上，这个菜单条是一个在外观与功能上都与标准菜单非常相似的工具条控件。这种菜单条不是具有顶级菜单项目，而是具有一系列纯文本按钮，用来显示一个下拉菜单。但是，作为一种特殊类型的工具条，菜单条具有某些标准菜单所不具有的功能，作为一个工具条控件：

- 可以使用标准工具条技术来定制菜单条，从而允许用户移动、删除或添加项目。
- 可以进行热跟踪，从而使得用户能够不必首先单击某个顶级项目，就能够知道鼠标当前所在的顶级项目位置。
- 菜单条可以被嵌入到Rebar控件中，从而具有以下功能：
- 可以有夹具，从而允许用户移动区段或改变区段的大小。
- 可以与其他区段在Rebar控件中共享一个条，例如，在前面的演示例子中与标准工具条共享一个条。
- 如果菜单条被邻接的区段所覆盖，可以显示一个人字形按钮，从而使得用户能够访问被隐

藏的项目。

- 可以有由应用程序所定义的背景位图。

这个文档讨论了如何实现菜单条，由于菜单条是工具条控件的一种特殊实现，因此重点将讨论菜单条所特有的特性。关于如何将工具条嵌入到Rebar控件中，请参见6.3节“创建Internet Explorer样式的工具条”。

6.4.1 在菜单条中加入工具条

如果需要在菜单条中加入工具条，应该实现以下步骤：

- 创建除了包含其他窗口样式之外还包含有TBSTYLE_FLAT样式的平面工具条。TBSTYLE_FLAT样式将打开热跟踪功能，在这个样式下，用户激活某个按钮之前，菜单条看起来都与普通的标准菜单完全一样。单击某个按钮之后，按钮将从工具条中突出显示出来，这就像标准的按钮一样。由于打开了热跟踪功能，需要激活一个按钮的动作实际上只要将鼠标移到这个按钮之上即可。如果鼠标移动到另一个按钮上，那么这个按钮将被激活，而原有的那个按钮则取消激活状态。
- 创建除了包含其他窗口样式之外还包含有TBSTYLE_LIST样式的列表按钮，这个样式将创建一个更窄的按钮，从而看起来更像标准的顶级菜单项目。
- 将按钮TBUTTON数据结构中的iBitmap数据成员设置为L_IMAGENONE，同时将其中的iString数据成员设置为按钮的文本，从而使得按钮只显示文本。
- 为每个按钮设置BTNS_DROPDOWN样式。当按钮被单击之后，工具条控件将向应用程序发送一条TBN_DROPDOWN通告消息，请求应用程序显示按钮的菜单。
- 将菜单条嵌入到一个Rebar区段中。打开夹具功能与人字形按钮显示功能，这些在6.3节“创建Internet Explorer样式的工具条”中进行了说明。
- 实现一个TBN_DROPDOWN处理程序，用来显示按钮被单击之后应该出现的下拉菜单。这个下拉菜单是一种弹出菜单，可以利用TrackPopupMenu函数创建这个菜单，并且菜单的左上角与按钮的左下角对齐。
- 实现键盘导航功能，从而使得在不用鼠标的情况下能够对菜单条进行完全操作。
- 实现菜单热跟踪功能。对于标准菜单，一旦某个顶级菜单项目的菜单内容被显示，那么如果将鼠标移动到另一个顶级菜单项目上，将会自动地显示新的项目菜单，同时消除原有项目的菜单。工具条控件能够对鼠标进行热跟踪，并且改变按钮的图像，同时能够自动显示新菜单。因此，应用程序必须实现这种特性。

上面这些步骤的实现比较简单，在本书的其他地方分别有相关的说明。最后两个步骤，即键盘导航与菜单热跟踪，将在后面的文档中进行讨论。

6.4.2 处理菜单热跟踪功能关闭之后的导航问题

用户可以选择利用鼠标、键盘、或者两者都使用来进行菜单条的导航。为实现菜单条导航功能，应用程序必须处理工具条通告消息并且监视鼠标与键盘。这个工作可以分为两个独立的

部分：带有菜单热跟踪功能的导航与没有菜单热跟踪功能的导航。本节将讨论在没有菜单被显示并且关闭了菜单热跟踪功能的情况下，进行菜单导航的问题。

1. 鼠标导航

如果菜单热跟踪功能被关闭，就可以将菜单条作为普通的工具条进行处理。如果用户利用鼠标进行导航，那么应用程序就必须处理TBN_DROPDOWN通告消息。当这个通告消息被接收之后，应该显示适当的下拉菜单，并且打开菜单热跟踪功能。此后，实现过程就与菜单热跟踪功能的实现一样，这在6.4.1节中进行了讨论。

正如在下面的“混合导航”中所讨论的那样，还必须处理TBN_HOTITEMCHANGE通告消息，从而保持对活跃按钮的跟踪。如果用户仅仅利用鼠标进行导航，那么这个通告消息就不会发生作用；但是，在进行鼠标与键盘的混合导航时，就需要处理这个通告消息。

2. 键盘导航

正如前面一节中所说的那样，在关闭菜单热跟踪功能之后，可以使用键盘进行一系列的导航操作。只有在工具条控件已经获取了焦点之后，工具条控件才能够支持键盘导航。同时，键盘导航并不能处理对菜单条所需的所有按键操作。处理键盘导航的最简单解决方案就是明确地处理相应的按键事件。

如果关闭了菜单热跟踪功能，那么应用程序就必须按照以下方法来处理键盘按键事件：

- 如果按下了F10键，利用TB_SETHOTITEM激活第一个按钮。
- 如果按下了向左箭头键与向右箭头键，利用TB_SETHOTITEM激活邻接按钮。如果用户试图导航出菜单条，那么就激活菜单条另一端的第一个按钮。
- 如果按下了ESCAPE键，利用TB_SETHOTITEM取消对活跃按钮的激活。这时，菜单条应该返回到第一个按钮被激活之前的状态。
- 如果按下了Alt+Key加速键，使用TB_MAPACCELERATOR消息来判断那个按钮与Key字符相对应，显示这个按钮的下拉菜单并且打开菜单热跟踪功能。
- 如果按下了向下箭头键，如果某个按钮被激活但没有显示任何菜单，那么就显示按钮的菜单并且打开菜单热跟踪功能。
- 如果按下了ENTER键，利用TB_SETHOTITEM取消对活跃按钮的激活。这时，菜单条应该返回到第一个按钮被激活之前的状态。

3. 混合导航

前面所列出的键盘导航处理程序能够完成两个基本的任务：对活跃按钮进行跟踪以及在按钮被选中之后显示适当的菜单。只要用户仅仅使用键盘进行导航，那么这些处理程序就足以完成相应的导航功能。但是，用户经常会结合使用键盘导航与鼠标导航。例如，用户可能会利用F10键激活第一个按钮，使用鼠标激活不同的按钮，然后利用向下箭头键打开菜单。如果仅仅监视键盘按键来跟踪活跃按钮，那么就会显示错误的菜单。因此，必须同时处理TBN_HOTITEMCHANGE通告消息，从而精确地对活跃按钮进行跟踪。

6.4.3 处理打开了菜单热跟踪功能的导航问题

一旦打开了菜单热跟踪功能，应用程序就必须改变自己对用户导航的响应方式。为了实现

标准菜单的行为，必须在应用程序中明确实现以下功能。

对于鼠标导航：

- 如果用户将鼠标移动到另一个按钮上，那么相应的菜单应该立刻出现，而先前菜单消失。
- 菜单热跟踪功能一直保持活跃，直到用户选择了一个命令，或者单击了不是菜单区域一部分的一个点。上面两个动作都会导致菜单热跟踪状态的关闭，直到单击了另一个按钮。
- 如果鼠标移出了菜单，那么当前下拉菜单仍然保持；直到鼠标重新回到菜单上，或者用户单击了菜单覆盖区域之外的一个点。如果鼠标返回到不同于当前被显示按钮的另一个按钮上，那么原有的菜单将被关闭，而新的按钮方法菜单则显示出来。

对于键盘导航：

- 如果按下了向右箭头键，那么，如果项目有一个子菜单，那么就显示这个子菜单。如果项目没有子菜单，那么就关闭这个菜单与任何子菜单，并且利用TB_SETHOTITEM激活下一个按钮，然后显示邻接按钮的菜单。如果激活了最后一个按钮，并且接收到这个消息，那么就显示第一个按钮的菜单。
- 如果按下了向左箭头键，那么，如果项目是一个子菜单，就关闭这个子菜单并且将焦点返回给父菜单。如果项目不是一个子菜单，那么就关闭这个菜单，并且利用TB_SETHOTITEM激活下一个按钮，并且显示这个按钮的菜单。
- 按下向左箭头键之后，将把焦点转移到左边。
 - 如果高亮显示的菜单项目在主菜单上，那么主菜单将被关闭，并且邻接按钮的菜单被显示。如果活跃按钮在工具条的最左端，那么将显示最后一个按钮的菜单。
 - 如果被高亮显示的菜单项目在一个子菜单上，那么子菜单将被关闭，并且将焦点交还给父菜单。
- 按下向右箭头键之后，将把焦点转移到右边。
 - 如果被高亮显示的菜单项目没有子菜单，那么将显示邻接按钮的菜单。如果活跃按钮在工具条的最右端，那么最后一个按钮的菜单将被显示。
 - 如果被高亮显示的菜单项目有一个子菜单，那么这个子菜单将被显示。
- 按下ESCAPE键之后，将返回到显示的上一步。
 - 如果子菜单被显示，那么将关闭这个子菜单，并且将焦点返回父菜单。
 - 如果按钮的菜单被显示，则关闭这个菜单，并且关闭菜单热跟踪功能。但项目的按钮仍然处于活跃状态。
 - 如果没有显示任何菜单，但按钮处于活跃状态，则关闭按钮的活跃状态并且关闭菜单热跟踪功能。
- 向上箭头键与向下箭头键仅仅用来在特殊的菜单内导航。
- ENTER键将导致与菜单项目相关联的命令得到执行。如果菜单项目有子菜单，那么ENTER键将显示这个子菜单。

与菜单热跟踪功能关闭的情况一样，应用程序必须处理鼠标导航、键盘与混合导航。但是，菜单被显示就意味着消息应该被不同地处理。

1. 菜单热跟踪的消息处理

菜单热跟踪功能要求在任何时候（除了切换到新菜单那一瞬间之外）都应该显示一个菜单，但是，被TrackPopupMenu所显示的下拉菜单是模态的。应用程序必须继续接收某些消息，包括WM_COMMAND、TBN_HOTITEMCHANGE以及与普通菜单相关联的消息（例如WM_MENUSELECT）。但是，应用程序可以不直接接收低级键盘消息或鼠标消息。为了处理消息（例如，WM_MOUSEMOVE），必须设立一个消息处理例程，用来对发送给菜单的消息进行解释。

当显示一个下拉菜单之后，就应该调用SetWindowHookEx，同时将idHook参数设置为WH_MSGFILTER，从而设置消息处理例程。所有发送给这个菜单的消息都将被传递给这个消息处理例程。例如，下面的代码段就可以用来设置一个消息处理例程，它将捕获所有到达下拉菜单的消息。其中，MsgHook是消息处理例程的名称，hhookMsg是这个例程的句柄。

```
hookMsg = SetWindowsHookEx(WH_MSGFILTER, MsgHook, HINST_THISDLL, 0);
```

通过将消息处理例程中的nCode参数设置为MSGF_MENU，就能够对菜单消息进行标识。其中，lParam参数将指向消息的MSG数据结构。关于哪些消息应该被处理以及如何对这些消息进行处理的详细信息，将在后面的文档中进行讨论。

应用程序应该调用CallNextHookEx，将所有的消息传递给下一个消息处理例程。这样做的目的是能够确保工具条控件接收到需要用来自按钮进行热跟踪的鼠标消息。

在新按钮被激活之后，应用程序必须利用WM_CANCELMODE消息关闭原有的下拉菜单，并且显示新菜单。同时，如果由于键盘导航或单击了菜单区之外的某个点，导致焦点转移，应用程序也必须关闭下拉菜单。只要关闭了一个菜单，就应该利用UnhookWindowsHookEx释放相应的菜单消息。如果需要显示另一个下拉菜单，那么就应该重新创建一个新的消息处理例程。当一个命令被激活运行之后，菜单将自动地被关闭，但必须明确地释放消息处理例程。

下面的代码段可以用来释放先前例子中所创建的消息处理例程：

```
UnhookWindowsHookEx(hhookMsg);
```

2. 鼠标导航

当一个普通的工具条控件对按钮进行热跟踪时，它将高亮显示活跃按钮并且在新的按钮被激活之后，向应用程序发送一条TBN_HOTITEMCHANGE通告消息。应用程序负责显示适当的下拉菜单，必须：

- 处理TBN_HOTITEMCHANGE通告消息，从而跟踪活跃按钮。当活跃按钮发生改变时，应该关闭原有的菜单，并且创建一个新菜单。
- 在按钮被单击之后，处理所发送的TBN_DROPDOWN通告消息。然后，应该关闭这个菜单，并且关闭菜单热跟踪功能。但按钮仍然维持活跃状态。
- 在消息处理例程中处理WM_LBUTTONDOWN、WM_RBUTTONDOWN与WM_MOUSEMOVE消息，从而跟踪鼠标的位置。如果在菜单区域之外单击了鼠标，那么就关闭当前的下拉菜单，取消热跟踪功能，将菜单条返回到被激活之前的状态。
- 如果激活了某个菜单命令，那么就关闭菜单热跟踪功能。这时，菜单将自动地被关闭，但必须明确地释放消息处理例程。

此外，还必须处理与菜单相关的消息，例如，在菜单项目需要显示一个子菜单时所发送的WM_INITMENUPOPUP消息。

3. 键盘导航

应用程序必须处理消息处理例程中的键盘消息，并且对那些影响菜单热跟踪特性的键盘消息进行处理。概括来说，有下面的键盘按键事件需要被处理：

- 如果按下了ESCAPE键，将把显示返回到上一级。如果显示的是子菜单，那么就必须关闭这个显示的子菜单。如果显示的是按钮的主菜单，就必须关闭这个主菜单并且关闭菜单热跟踪功能。但按钮仍然维持活跃状态。
- 如果按下了向右箭头键，那么，如果项目有子菜单，则显示这个子菜单。如果项目没有子菜单，则关闭这个菜单与任何子菜单，并且利用TB_SETHOTITEM激活下一个按钮，并且显示这个按钮的菜单。如果在接收这个通告消息时，最后一个按钮处于活跃状态，则显示第一个按钮的菜单。
- 如果按下了向左箭头键，那么，如果项目在一个子菜单中，则关闭这个子菜单并且将焦点转移到父菜单。如果项目不是子菜单，则关闭这菜单，并且利用TB_SETHOTITEM激活邻接的按钮，并且显示这个按钮的菜单。如果在接收到这个通告消息时，第一个按钮处于活跃状态，则显示最后一个按钮的菜单。
- 如果按下了向上箭头键或向下箭头键，那么这些键将用来在菜单内部进行导航，但不会直接影响菜单热跟踪功能。
- 如果按下了Alt+Key加速键，就应该使用TB_MAPACCELERATOR消息来判断哪个按钮与Key字符相对应。如果Key字符与不是活跃按钮的另一个按钮相对应，那么就应该关闭当前下拉菜单，并且利用TB_SETHOTITEM激活新按钮，并且显示这个按钮的菜单。如果Key字符与当前被显示的按钮相对应，那么就关闭下拉菜单并且取消菜单的热跟踪功能。但这个按钮仍然维持活跃状态。

6.5 通用控件的本地化支持

通用控件能够提供对国际语言的内置支持，这些功能有助于简化本地化应用程序的实现。

6.5.1 为通用控件指定一种语言

如果需要为通用控件指定一种不同于系统语言的语言，就应该调用InitMUILanguage函数。由这个函数所指定的语言只能应用于调用这个函数的处理程序。

如果需要判断通用控件目前所使用的语言，就应该调用GetMUILanguage函数，这个函数将返回由先前对InitMUILanguage函数调用时所设置的值。同时，这个函数返回调用这个函数所使用的进程所使用的语言。如果没有调用InitMUILanguage函数，或者从另一个进程调用这个函数，那么GetMUILanguage将返回缺省值。

6.5.2 在对话框中为控件指定一种语言

与通用控件不同，预先定义的控件（例如按钮或编辑框）在缺省情况下并不使用当前系统语言。本地字体控件是一个不可见的控件，这个控件在后台工作，从而允许对话框的预定义控件显示当前系统语言。

如果需要使用本地字体控件, 应该:

1) 调用InitCommonControlsEx函数, 初始化本地字体控件。将lpInitCtrls所指向的INITCOMMONCONTROLSEX数据结构中的dwICC数据成员设置为ICC_NATIVEFNTCTL_CLASS。

2) 将这个控件添加到对话框的资源脚本中。然后, 设置以下一个或多个样式标志位, 从而指定受影响的控件:

标 志 位	所应用的控件
NFS_ALL	所有控件
NFS_BUTTON	按钮控件
NFS_EDIT	编辑框控件
NFS_LISTCOMBO	列表控件、组合框控件、列表视图控件以及扩展组合框控件
NFS_STATIC	静态控件
NFS_USEFONTASSOC	这个控件将使用字体相关性功能, 而不是切换到本地字体。 这个标志位仅仅对于远东操作平台有效。所有的其他平台都应该忽略这个标志位

下面的例子演示了如何向资源脚本中添加本地字体控件, 它将导致对话框的编辑控件、列表控件与组合框控件使用当前系统语言显示文本。

```
CONTROL  "*", -1, "NativeFontCtl", NFS_EDIT | NFS_LISTCOMBO, 0, 0, 0, 0
```

第7章 通用 API

7.1 通用控件窗口类

通用控件库中给出了以下窗口类名：

ANIMATE_CLASS	创建动画控件。这些控件可用来显示AVI剪辑。
DATETIMEPICK_CLASS	创建日期与时间检出器控件。这些控件能够提供一个简单而具有指导性的界面，用来与用户交换日期与时间信息。
HOTKEY_CLASS	创建热键控件。这些控件能够使得用户更为方便地定义热键。
MONTHCAL_CLASS	创建月历控件。这些控件能够为用户提供一个简单而具有指导性的方法，用来从一个比较熟悉的界面上选择日期。
PROGRESS_CLASS	创建进度条。在一个耗时很长的操作过程中，这些控件可用来显示操作进度。
REBARCLASSNAME	创建Rebar控件。这些控件可以作为子窗口的容器。
STATUSCLASSNAME	创建状态窗口。这些控件可以在一个水平的窗口中显示状态信息。
TOOLBARCLASSNAME	创建工具条。这些控件中包含有用来执行菜单命令的按钮。
TOOLTIPS_CLASS	创建工具提示控件。这些控件能够显示包含有一行文本信息的小弹出窗口，用来描述应用程序中某个工具的作用。
TRACKBAR_CLASS	创建轨迹条。这些控件能够让用户通过移动滑动触头的方法来选择范围值。
UPDOWN_CLASS	创建增减数控件。这些控件由一对箭头与一个编辑控件组成。通过单击某个箭头，能够使得编辑控件中的值自增或者自减。
WC_COMBOBOXEX	创建扩展组合框控件。这些控件对组合框控件进行了扩展，从而能够为项目图像提供本地支持。
WC_HEADER	创建头标控件。这些控件在信息的列顶端显示头文件信息，并且能够让用户通过单击头文件方法进行信息的排序。
WC_IPADDRESS	创建IP地址控件。这些控件与编辑控件类似，但它们允许用户以Internet协议（IP）格式输入数字地址。
WC_LISTVIEW	创建列表视图控件。这些控件能够显示项目的集合，每个项目中包含有一个图标与一个标号，并且这些控件能够提

WC_PAGESCROLLER	供对项目进行排列的几种方法。
WC_TABCONTROL	创建Pager控件。这些控件用来包含与滚动另一个窗口。
	创建选项卡控件。这些控件能够为一个窗口或对话框上的相同区域定义多个页面。每个页面中包含有一个信息集合或一组控件，当用户选择相应的选项卡时，应用程序将显示这些信息或控件组。
WC_TREEVIEW	创建树视图控件。这些控件能够显示关于项目的层次性列表，其中每个项目都包含有一个标号与一个可选的位图。

7.2 通用控件样式

下面给出了通用的控件样式。除非特别声明，否则这些样式将适用于头标控件、工具条控件与状态条窗口。

CCS_ADJUSTABLE	打开工具条的内置自定义特性，从而允许用户将一个按钮拖动到新的位置；或者，通过将按钮移出工具条来删除此按钮。此外，用户可以缩短工期工具条，从而显示Customize Toolbar（自定义工具条）对话框，这个对话框允许添加、删除与重新排列工具条按钮。
CCS_BOTTOM	使控件定位于父窗口客户区的底部，应将控件的宽度设置为与父窗口的宽度相同。缺省情况下，状态窗口具有此样式。
CCS_LEFT	在4.70版本中有效。它将会使得控件在父窗口的左边竖直地显示。
CCS_NODIVIDER	这个样式将会使得不会在控件的顶部显示两个像素宽度的高亮线条。
CCS_NOMOVEX	在4.70版本中有效。在响应WM_SIZE消息时，这个样式将会使得控件重新调整自身大小，并且在竖直方向上移动，但不会在水平方向上移动。如果CCS_NORESIZE有效，这个样式将不会发生作用。
CCS_NOMOVEY	在响应WM_SIZE消息时，这个样式将会使得控件重新调整自身大小，并且在水平方向上移动，但不会在竖直方向上移动。如果CCS_NORESIZE有效，这个样式将不会发生作用。缺省情况下，头文件窗口使用这个样式。
CCS_NOPARENTALIGN	阻止控件自动地移动到父窗口的顶部或底部。相反，即使是父窗口的大小发生了改变，这些控件仍然保持在父窗口中的相应位置。如果使用了CCS_TOP与CCS_BOTTOM，那么高度将调整到缺省值，但控件的位置与宽度保持不变。
CCS_NORESIZE	阻止控件在设置自己的初始大小或新大小时使用缺省宽度

CCS_RIGHT	与高度。相反，控件将使用在创建或改变大小请求中所指定的宽度高度。
CCS_TOP	在4.70版本中有效。这个样式将使得控件在父窗口的右方被竖直地显示。
CCS_VERT	这个样式将使得控件定位于父窗口客户区的顶部，并且将控件的宽度设置为与父窗口宽度相同。缺省情况下，工具条将使用这个样式。
	在4.70版本中有效。这个样式将使得控件按照竖直的方式进行显示。

7.3 通用API参考

7.3.1 通用API函数

GetEffectiveClientRect

计算客户区矩形的面积。

```
void GetEffectiveClientRect(
    HWND hWnd ,
    LPRECT lprc ,
    LPINT lpInfo
);
```

参数

hWnd: 客户区矩形所在窗口的句柄。

lprc: 接收矩形面积的RECT数据结构地址。

lpInfo: 整型数组的地址，用来标志客户区的控件。每个控件都要求有一对连续的元素，其中第一个元素必须为非零，而第二个元素必须是控件的标识号。数组中的最后一个元素对必须是0，用来说明数组的结尾。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

GetMUILanguage

返回通用控件为某个特定的进程在当前所使用的语言。

LANGID GetMUILanguage (VOID) ;

参数

无。

返回值

通过调用InitMUILanguage, 返回某个应用程序为通用控件所指定的语言LANGID。GetMUILanguage可以返回调用进程的值, 如果InitMUILanguage没有被调用或者不是从相同的进程中被调用, InitMUILanguage将返回中间语言的LANGID, MAKELANGID (LANG_NEUTRAL, SBULANG_NETRAL)。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

InitCommonControls

注册与初始化通用控件窗口类。这个函数正在被舍弃; 因此, 在新的应用程序中应该使用InitCommonControlsEx函数。

void InitCommonControls (VOID) ;

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

InitCommonControlsEx

在通用控件的动态链接库中注册特定的通用控件类。

```
BOOL InitCommonControlsEx (  
    LPINITCOMMONCONTROLSEX lpInitCtrls  
) ;
```

参数

lpInitCtrls: INITCOMMONCONTROLSEX数据结构的地址, 在这个数据结构中包含了用来说明将要对哪个控件类进行注册的信息。

返回值

如果操作成功, 则返回TRUE; 否则, 返回FALSE。

说明

注意 每次对InitCommonControlsEx的调用所得到的效果具有累加性。例如, 如果InitCommonControlsEx首先利用ICC_UPDOWN_CLASS标志位进行了一次调用, 后来又利用ICC_HOTKEY_CLASS标志位进行了一次调用; 那么, 最终结果是增减数控件与热键控件都被注册, 并且都在这个应用程序中可用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

InitMUILanguage

允许应用程序为通用控件指定一种不同于系统语言的特定语言。

```
VOID InitMUILanguage (  
    LANGID uiLang  
);
```

参数

uiLang: 通用控件将使用语言的LANGID值。

返回值

没有返回值。

说明

这个函数允许应用程序覆盖系统语言设置, 并且为通用控件指定一种不同的语言。当然, 所选择的语言仅仅适用于InitMUILanguage的调用进程。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

参见

GetMUILanguage。

ShowHideMenuCtl

设置或删除指定菜单项目的对号标记属性, 并且显示或隐藏相应的控件。如果指定的菜单项目当前没有对号标记, 这个函数将给它添加一个对号标记, 然后显示相应的控件。如果指定的菜单项目已经存在一个对号标记, 函数将删除这个对号标记, 然后隐藏相应的控件。

```
BOOL ShowHideMenuCtl (
    HWND hWnd,
    UINT uFlags,
    LPINT lpInfo
);
```

参数

hWnd: 包含有菜单与控件的窗口句柄。

uFlags: 将要接收或丢失对号标记的菜单项目标识。

lpInfo: 保存数值对的数组地址。第一个数值对中的第二个值必须是应用程序主菜单的句柄。而后续的每个数值对中则包含有菜单项目的标识以及控件窗口的标识。函数将对这个数组进行搜索, 查到与uFlags相匹配的值。如果这个值被找到, 函数将对相应的菜单项目设置或取消对号标记, 并且显示或隐藏相应的控件。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

7.3.2 通用API消息

CCM_GETUNICODEFORMAT

CCM_GETUNICODEFORMAT消息获取Unicode字符格式的控件标志位。

```
CCM_GETUNICODEFORMAT
wParam = 0;
```

```
lParam = 0;
```

参数

这个消息没有参数。

返回值

返回关于控件的统一编码格式的标志位。如果这个值非零，那么表示控件正在使用的是Unicode字符；如果这个值为0，表示控件正在使用ANSI字符。

实例

以下函数可以用于Microsoft Windows 95或Microsoft Windows 98系统，测试某个属性页面控件是否支持Unicode。关于测试控件是否支持Unicode的更多信息，请参见下面的“说明”。

```
BOOL IsComctl32Unicode(void)
{
    PROPSHEETPAGEW pspW = { PROPSHEETPAGEW_V1_SIZE };
    HPROPSHEETPAGE hpage = CreatePropertySheetW(&pspW);
    if (hpage) {
        DestroyPropertySheetPage(hpage);
        return TRUE;
    }
    else {
        return FALSE;
    }
}
```

说明

Unicode格式的标志位信息在Microsoft Windows NT 4.71版本或更新版本的Comctl32.dll中使用。因此，这个消息能够被Windows 2000及其更新版本、带有Microsoft Internet Explorer 4.0及其更新版本的Windows NT 4.0所支持。但事实上，这个消息仅仅对于带有5.80版本或更新版本Comctl32.dll的Windows 95或Windows 98系统有用。这就意味着上述两个系统必须安装了Internet Explorer 5.0或更新版本，对于Windows 95或Windows 98系统中更早版本的Internet Explorer，系统将忽略Unicode格式的标志位；因此，这些标志位的值对于系统是否允许控件支持Unicode毫无影响。在这种情况下，就必须自行测试系统是否需要提供Unicode支持。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

CCM_SETUNICODEFORMAT。

CCM_GETVERSION

返回最新CCM_SETVERSION消息所设置的控件版本号。

```
CCM_GETVERSION
    wParam = 0;
    lParam = 0;
```

参数

无。

返回值

返回最新CCM_SETVERSION消息所设置的控件版本号。如果没有使用这个消息，将返回0。

说明

这个消息不能返回DLL版本号。关于如何使用DllGetVersion来获取当前DLL版本号的更多信息，请参见Shell Versions（Shell版本）。

注意 控件的版本号是基于控件自身进行设置的；因此，并非所有控件的版本号都相同。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CCM_SETUNICODEFORMAT

CCM_SETUNICODEFORMAT消息为控件设置Unicode字符格式的标志位。这个消息允许用户在运行时改变控件所使用的字符集，而不必重新创建控件。

```
CCM_SETUNICODEFORMAT
    wParam = (WPARAM)(BOOL) fUnicode;
    lParam = 0;
```

参数

fUnicode：这个值用来决定被控件所使用的字符集。如果这个值为TRUE，那么控件将使用Unicode字符；如果这个值为FALSE，控件将使用ANSI字符。

返回值

返回本控件的先前Unicode格式标志位。

说明

在带有4.71版本或更新版本Comctl32.dll的Microsoft Windows NT系统中使用Unicode格式的字符标志位。因此，这个消息能够被Windows 2000及其更新版本、带有Microsoft Internet Explorer 4.0及其更新版本的Windows NT 4.0所支持。但事实上，这个消息仅仅对带有5.80版本

或更新版本Comctl32.dll的Windows 95或Windows 98系统有用。这就意味着上述两个系统必须安装了Internet Explorer 5.0或更新版本,对于Windows 95或Windows 98系统中更早版本的Internet Explorer,系统将忽略Unicode格式的标志位,因此,这些标志位的值对于系统是否允许控件支持Unicode毫无影响。关于如何测试一个控件是否支持Unicode的更多信息,请参见CCM_GETUNICODEFORMAT。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

CCM_GETUNICODEFORMAT。

CCM_SETVERSION

这个消息用来告知控件,将要使用一个与特定控件版本相关的行为。

CCM_SETVERSION

```
wParam = (int) iVersion;  
lParam = 0;
```

参数

iVersion: 版本号。

返回值

返回在先前CCM_SETVERSION消息中所指定的版本号。如果iVersion所设置的版本号值大于当前DLL版本号,将返回-1。

说明

在少数情况下,根据通用控件版本的不同,某个控件可能会有不同的行为动作。之所以会出现这种情况,主要是因为后续的版本中修复了某些Bug。CCM_SETVERSION可以用来告知控件,用户需要使用控件的哪个版本所具有的行为。同时,也可以发送CCM_GETVERSION消息,来判断自己所指定的控件版本。

注意 这个消息仅仅对所发送的控件进行版本号的设置。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 5.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 5.0或更新版本的

Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

WM_NOTIFY

当发生了某个事件或控件需要某些信息时, 通用控件将向它的父窗口发送WM_NOTIFY消息。

```
WM_NOTIFY
    idCtrl = (int) wParam;
    pnmh = (LPNMHDR) lParam;
```

参数

idCtrl: 发送消息的通用控件的标识符。这个标识符不一定具有唯一性, 应用程序应该使用NMHDR数据结构(以lParam参数的形式进行传送)中的hwndFrom或idFrom成员来对控件进行标识。

pnmh: 指向NMHDR数据结构的指针, 在这个数据结构中包含有通告消息代码与其他一些更多信息。对于某些通告消息, 这个参数将指向以NMHDR数据结构作为第一个成员的更大数据结构。

返回值

除非通告消息进行了特别指定, 否则将忽略返回值。

说明

如果消息处理器在一个对话框过程中, 必须使用带有DWL_MSGRESULT参数的SetWindowLong函数来设置返回值。

标准的Windows控件(如编辑控件、组合框、列表框、按钮、滚动条以及静态控件)并不发送WM_NOTIFY消息。如果需要判断一个通用控件是否能够发送WM_NOTIFY消息, 以及将发送什么通告消息代码, 请参见相应控件的文档说明。

对于Windows 2000及其更新系统, WM_NOTIFY消息不能在进程之间发送。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在winuser.h中进行了声明。

WM_NOTIFYFORMAT

用来判断某个窗口是否接受WM_NOTIFY通告消息中的ANSI或Unicode数据结构。WM_NOTIFY消息能够从通用控件发送到这个通用控件的父窗口, 也能够从某个通用控件的父窗口发送到这个通用控件。

```
WM_NOTIFYFORMAT
```

```
hwndFrom = (HWND) wParam;  
Command = lParam;
```

参数

hwndFrom: 发送WM_NOTIFYFORMAT消息的窗口句柄。如果参数Command的值是NF_QUERY, 那么这个参数就是控件的句柄。如果参数Command是NF_REQUERY, 那么这个参数就是控件的父窗口句柄。

Command: 用来指定WM_NOTIFYFORMAT消息特性的命令值。这个参数的值将是以下之一:

- | | |
|------------|---|
| NF_QUERY | 表示这个消息是一个查询, 以便判断在WM_NOTIFY消息中应该使用ANSI结构或Unicode结构。在控件创建的过程中以及对NF_REQUERY命令进行响应时, 这个命令将从控件发送到控件父窗口中。 |
| NF_REQUERY | 表示这个消息是一个请求, 用来请求控件向控件的父窗口发送NF_QUERY形式的消息。因此, 这个命令是从父窗口发出的, 表示父窗口要求控件重新进行查询, 以便确定在WM_NOTIFY消息中所应该使用的数据结构类型。 |

返回值

这个函数将返回以下值之一:

- | | |
|-------------|-------------------------------------|
| NFR_ANSI | 表示控件在发送WM_NOTIFY消息时应该使用ANSI数据结构。 |
| NFR_UNICODE | 表示控件在发送WM_NOTIFY消息时应该使用Unicode数据结构。 |
| 0 | 表示在处理过程中发生了错误。 |

如果参数Command的值是NF_REQUERY, 返回值就是重新查询操作的结果。

说明

在创建一个通用控件之后, 这个通用控件将向自己的父窗口发送WM_NOTIFYFORMAT消息, 以便决定在WM_NOTIFY消息中应该使用的数据结构类型。如果父窗口没有处理这个消息, 那么DefWindowProc函数将根据父窗口的类型来对消息进行响应。也就是说, 如果父窗口是一个Unicode窗口, 那么DefWindowProc将返回NFR_UNICODE; 如果父窗口是一个ANSI窗口, DefWindowProc将返回NFR_ANSI。如果父窗口是一个对话框, 并且不能处理这些消息, 那么DefDlgProc函数将根据对话框的类型(Unicode或ANSI)作出相应的响应。

通过将参数WM_NOTIFY的值设置为lParam, 并且向控件发送一个NF_REQUERY消息的方法, 父窗口就可以改变通用控件在WM_NOTIFYFORMAT消息中所使用的数据结构类型。这样, 将会导致控件向父窗口发送NF_QUERY形式的WM_NOTIFYFORMAT消息。

所有的通用控件都能够发送WM_NOTIFYFORMAT消息; 但是, 标准的Windows控件(编辑控件、组合框、列表框、按钮、滚动条以及静态控件)却不能发送这种消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件：在winuser.h中进行了声明。

7.3.3 通用API宏

FORWARD_WM_NOTIFY

这个函数用来发送或转发WM_NOTIFY消息。

```
VOID FORWARD_WM_NOTIFY (
    hwnd,
    idFrom,
    pnmhdr,
    fn
);
```

参数

hwnd：接收WM_NOTIFY消息的窗口句柄。

idFrom：发送这个消息的控件的标识符。

pnmhdr：NMHDR数据结构的地址，在这个数据结构中包含有通告消息代码以及附加的信息。对于某些通告消息，这个参数将指向以NMHDR数据结构作为第一个成员的更大的数据结构。

fn：发送或转发WM_NOTIFY消息的函数。这个参数可以是SendMessage函数或PostMessage函数。

返回值

返回一个值，这个值的含义取决于fn参数。

说明

FORWARD_WM_NOTIFY宏的定义如下：

```
#define FORWARD_WM_NOTIFY (hwnd,idFrom,pnmhdr,fn) \
    (void)(fn)(hwnd),WM_NOTIFY,(LPARAM)(int)(id), \
    (LPARAM)(NMHDR FAR*)(pnmhdr))
```

环境需求

Windows NT/2000：需要使用Windows NT 3.1或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

HANDLE_WM_NOTIFY

调用一个函数，这个函数用来处理WM_NOTIFY消息。

```
HANDLE_WM_NOTIFY (
    hwnd,
    wParam,
    lParam,
```

```
fn
);
```

参数

hwnd: 接收WM_NOTIFY消息的窗口句柄。

wParam: WM_NOTIFY消息的第一个参数。

lParam: WM_NOTIFY消息的第二个参数。

fn: 处理WM_NOTIFY消息的函数。

返回值

返回一个值, 这个值的含义取决于fn参数。

说明

HANDLE_WM_NOTIFY宏的定义如下:

```
#define HANDLE_WM_NOTIFY (hwnd,wParam,lParam,fn) \
    (fn)((hwnd), (int)(wParam), (NMHDR FAR*)(lParam))
```

环境需求

Windows NT/2000: 需要使用Windows NT 3.1或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

INDEXTOSTATEIMAGEMASK

准备好关于状态图像的索引, 以便树视图控件或列表视图控件可以使用这个索引来恢复某个项目的状态图像。

```
UINT INDEXTOSTATEIMAGEMASK (
    UINT i
);
```

参数

i: 状态图像的索引号。

说明

INDEXTOSTATEIMAGEMASK宏的定义如下:

```
#define INDEXTOSTATEIMAGEMASK(i)((i) <= 12)
```

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

7.3.4 通用API通告消息

NM_CHAR

当一个字符键被处理时，控件将发送一个NM_CHAR通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CHAR
    lpnmc = (LPNMCHAR) lParam;
```

参数

lpnmc: 指向NMCHAR数据结构的指针，在这个数据结构中包含有关于引起通告消息发生的字符的更多信息。

返回值

对于大多数控件，返回值将被忽略。关于更多信息，请参见相应控件的文档说明。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

NM_CHAR (在工具条控件中)。

NM_CLICK

这个通告消息用来告知控件的父窗口，用户在这个控件上单击了鼠标左键。NM_CLICK是以WM_NOTIFY消息的形式进行发送的。

```
NM_CLICK
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

这个控件将忽略所得到的返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在winuser.h中进行了声明。

NM_DBLCLK

这个通告消息用来告知控件的父窗口，用户在这个控件上双击了鼠标左键。NM_DBLCLK是以WM_NOTIFY消息的形式进行发送的。

```
NM_DBLCLK  
lpmh = (LPNMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

这个控件将忽略函数的返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在winuser.h中进行了声明。

NM_HOVER

当鼠标移动到某个项目之上时，控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_HOVER  
lpmh = (LPNMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

除非进行了特别的指定，否则将返回0，从而允许控件正常地处理鼠标悬停动作，或者，返回非零值，从而阻止鼠标悬停动作被处理。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

NM_KEYDOWN

当控件拥有键盘焦点并且用户按下了某个键时，控件将会发送这个通告消息。这个通告消

息是以WM_NOTIFY消息的形式进行发送的。

```
NM_KEYDOWN
lpmk = (LPNMKEY) lParam;
```

参数

lpmk: NMKEY数据结构的地址, 在这个数据结构中包含有关于引起这个通告消息产生的按键的更多信息。

返回值

如果需要阻止控件处理这个按键动作, 那么就应该返回非零值; 否则, 就应该返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NM_KILLFOCUS

告知控件的父窗口, 控件已经失去了输入焦点。NM_KILLFOCUS通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_KILLFOCUS
lpmh = (LPNMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于这个通告消息的更多信息。

返回值

控件将忽略返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在winuser.h中进行了声明。

NM_NCHITTEST

当控件接收到一个WM_NOTIFY_NCHITTEST消息时, 它将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_NCHITTEST
lpmmouse = (LPNM_MOUSE) lParam;
```

参数

lpnmouse: NM_MOUSE数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。其中的pt成员包含有鼠标击中测试消息中所得到的鼠标坐标。

返回值

除非进行了特别的指定, 否则将返回0, 从而允许控件执行鼠标击中测试消息中的缺省处理动作, 或者, 返回在WM_NCHITTEST中所定义的HT*值, 从而覆盖缺省的鼠标击中测试处理过程。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NM_OUTOFMEMORY

告知控件的父窗口, 由于没有足够的可用空间, 控件不能完成某个操作。NM_OUTOFMEMORY通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_OUTOFMEMORY  
lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略所得到的返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RCLICK

告知控件的父窗口, 用户在控件上单击了鼠标右键。NM_RCLICK通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RCLICK  
lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略所得到的返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RDBLCLK

告知控件的父窗口, 用户在控件上双击了鼠标右键。NM_RDBLCLK通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RDBLCLK
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有与本通告消息的更多信息。

返回值

控件将忽略所得到的返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RELEASEDCAPTURE

告知控件的父窗口, 控件正在释放鼠标捕获的内容。这个通告消息是以WM_NOTIFY消息形式发送的。

```
NM_RELEASEDCAPTURE
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

除非进行了特别的指定, 否则控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RETURN

告知控件的父窗口, 控件目前拥有输入焦点并且用户已经按下了ENTER键。NM_RETURN通告消息是以WM_NOTIFY通告消息形式进行发送的。

NM_RETURN

```
lpmh = (LPNMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略所得到的返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_SETCURSOR

告知控件的父窗口, 为了对NM_SETCURSOR消息进行响应, 控件正在设置光标。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

NM_SETCURSOR

```
lpmmm = (LPNM_MOUSE) lParam;
```

参数

lpmmm: NM_MOUSE数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

除非进行了特别的指定, 否则将返回非零值, 从而允许控件设置光标; 否则, 返回0, 从而阻止控件进行光标的设置。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_SETFOCUS

告知控件的父窗口, 控件已经接收到输入焦点。NM_SETFOCUS通告消息是以WM_NOTIFY消息的形式进行发送的。

NM_SETFOCUS

```
lpmnh = (LPNMHDR) lParam;
```

参数

lpmnh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略所得到的返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_TOOLTIPSCREATED

告知控件的父窗口, 控件已经创建了一个工具提示控件。这个通告消息是以WM_NOTIFY消息形式进行发送的。

语法

NM_TOOLTIPSCREATED

```
lpmnttc = (LPNMHDR) lParam;
```

参数

lpmnttc: NMTOOLTIPSCREATED数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

除非进行了特别的指定, 否则控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

7.3.5 通用API数据结构

COLORSCHEME

包含有工具条或Rebar控件中按钮的绘制信息。

```
typedef struct tagCOLORSCHEME{
    DWORD dwSize;
    COLORREF clrBtnHighlight;
    COLORREF clrBtnShadow;
}COLORSCHEME, *LPCOLORSCHEME;
```

成员

dwSize: 这个数据结构的大小, 以字节为单位。

clrBtnHighlight: COLORREF的值, 用来表示按钮的高亮颜色。在这里, 使用CLR_DEFAULT作为缺省高亮颜色。

clrBtnShadow: COLORREF的值, 用来表示按钮的阴影颜色。在这里, 使用CLR_DEFAULT作为缺省的阴影颜色。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

INITCOMMONCONTROLSEX

在这个数据结构中包含有用从动态链接库 (DLL) 中加载通用控件类的信息。这个数据结构通常被InitCommonControlsEx函数所使用。

```
typedef struct tagINITCOMMONCONTROLSEX {
    DWORD dwSize;
    DWORD dwICC;
} INITCOMMONCONTROLSEX, *LPINITCOMMONCONTROLSEX;
```

成员

dwSize: 这个数据结构的大小, 以字节为单位。

dwICC: 用来设置位标志, 这些位标志用来说明将要 DLL 中加载哪个通用控件类。这个参数只可以是以下值的组合:

ICC_ANIMATE_CLASS

加载动画控件类。

ICC_BAR_CLASSES

加载工具条、状态条、轨迹条以及工具提示控件类。

ICC_COOL_CLASSES

加载Rebar控件类。

ICC_DATE_CLASSES	加载日期与时间检出器控件类。
ICC_HOTKEY_CLASS	加载热键控件类。
ICC_INTERNET_CLASSES	加载IP地址类。
ICC_LISTVIEW_CLASSES	加载列表视图以及头标控件类。
ICC_PAGESCROLLER_CLASS	加载Pager控件类。
ICC_PROGRESS_CLASS	加载进度条控件类。
ICC_TAB_CLASSES	加载选项卡以及工具提示控件类。
ICC_TREEVIEW_CLASSES	加载树视图以及工具提示控件类。
ICC_UPDOWN_CLASS	加载增减数控件类。
ICC_USEREX_CLASSES	加载扩展组合框类。
ICC_WIN95_CLASSES	加载动画控件类、头标控件类、热键控件类、列表视图控件类、进度条控件类、状态条控件类、选项卡控件类、工具提示控件类、工具条控件类、轨迹条控件类、树视图控件类以及增减数控件类。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMCHAR

在这个数据结构中包含有字符通告消息中所使用的信息。

```
typedef struct tagNMCHAR{
    NMHDR hdr;
    UINT ch;
    DWORD dwItemPrev;
    DWORD dwItemNext;
}NMCHAR, FAR *LPNMCHAR;
```

成员

hdr: 包含有关于本通告消息更多信息的NMHDR数据结构。

ch: 正在被处理的字符。

dwItemPrev: 正在发送这个通告消息的控件所决定的一个32位值。

dwItemNext: 正在发送这个通告消息的控件所决定的一个32位值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMHDR

在这个数据结构中包含有关于某个通告消息的信息。

```
typedef struct tagNMHDR {  
    HWND hwndFrom;  
    UINT idFrom;  
    UINT code;  
} NMHDR;
```

成员

hwndFrom: 正在发送消息的控件的窗口句柄。

idFrom: 正在发送消息的控件的标识符。

code: 通告消息代码。这个数据成员可以是与特定控件相关的通告消息代码, 也可以是一个通用的通告消息代码。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在winuser.h中进行了声明。

NMKEY

在这个数据结构中包含有按键通告消息中所使用的信息。

```
typedef struct tagNMKEY {  
    NMHDR hdr;  
    UINT nVKey;  
    UINT uFlags;  
} NMKEY, FAR *LPNMKEY;
```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的更多信息。

nVKey: 导致这个事件发生的按键的虚拟按键代码。

uFlags: 与特定按键相关联的标识。这些标识与在WM_KEYDOWN消息的lParam参数中的高位字的标识相同。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_MOUSE

在这个数据结构中包含有鼠标通告消息所使用的信息。

```
typedef struct tagNM_MOUSE {
    NMHDR hdr;
    DWORD dwItemSpec;
    DWORD dwItemData;
    POINT pt;
    LPARAM dwHitInfo;
} NM_MOUSE, FAR* LPNM_MOUSE;
```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的更多信息。

dwItemSpec: 与特定控件相关联的项目标识符。

dwItemData: 与特定控件相关联的项目数据。

pt: POINT数据结构, 在这个数据结构中包含有在鼠标单击时鼠标所在的屏幕坐标。

dwHitInfo: 在这个数据成员中包含有关于鼠标指针在项目或控件上位置的信息。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NM_OBJECTNOTIFY

在这个数据结构中包含有被TBN_GETOBJECT、TCN_GETOBJECT以及PSN_GETOBJECT通告消息使用的信息。

```
typedef struct tagNM_OBJECTNOTIFY {
    NMHDR hdr;
    int iitem;
    IID* piid;
```

```

IUnknown* pObject;
HRESULT hResult;
} NM_OBJECT_NOTIFY, FAR* LPNM_OBJECT_NOTIFY;

```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的更多信息。

item: 与特定控件相关联的项目标识符。这个数据成员的值将遵守发送这个通告消息的控件的项目标识标准。但是, 这个数据成员不能在PSN_GETOBJECT通告消息中所使用。

pid: 被请求对象的接口标识符。

pObject: 正在处理通告消息的窗口所提供的对象的地址, 处理这个通告消息的应用程序将会对这个数据成员进行设置。

hResult: COM的成功或失败标志位, 处理这个通告消息的应用程序将会对这个数据成员进行设置。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMTOOLTIPS_CREATED

在这个数据结构中包含有被NM_TOOLTIPSCREATED通告消息所使用的信息。

```

typedef struct tagNM_TOOLTIPSCREATED {
    NMHDR hdr;
    HWND hwndToolTips;
} NM_TOOLTIPSCREATED, *LPNM_TOOLTIPSCREATED;

```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的更多信息。

hwndToolTips: 所创建的工具提示控件的窗口句柄。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows NT 4.0)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

第8章 定制控件外观

定制绘图 (custom draw) 并不是一个通用控件, 而是许多通用控件都提供的一种服务。在自定义控件的外观方面, 定制绘图服务能够为应用程序提供极大的灵活性。利用定制绘图通告消息, 可以非常方便地在应用程序中改变用来显示项目的字体, 或者在不必进行完全定制绘图操作的情况下, 手工进行控件外观的设计。

8.1 关于定制绘图

本节将讨论关于定制绘图功能的基础知识, 并且从概念上说明应用程序如何支持定制绘图。目前, 以下控件能够支持定制绘图功能:

- 头标控件
- 列表视图控件
- Rebar控件
- 工具条控件
- 工具提示控件
- 轨迹条控件
- 树状视图控件

注意 定制绘图功能在Comctl32.dll的4.70版本及其后续版本中实现。

8.1.1 关于定制绘图通告消息

所有能够支持定制绘图功能的通用控件都会在外观绘制操作过程的某个特定阶段发送NM_CUSTOMDRAW通告消息。这些通告消息描述了能够作用于整个控件的控件外观绘制操作, 同时也描述了仅仅作用于控件中某个特定项目的控件外观绘制操作。与许多通告消息相似, NM_CUSTOMDRAW通告消息也是以WM_NOTIFY消息的形式进行发送的。

定制绘图通告消息中的lParam参数可以是NMCUSTOMDRAW数据结构的地址, 也可以是包含有以NMCUSTOMDRAW数据结构作为其第一个数据成员的特定控件数据结构。下表说明了控件以及控件所使用的数据结构之间的关系:

数据结构	数据结构的使用者
NMCUSTOMDRAW	Rebar控件、轨迹条控件以及头标控件
NMLVCUSTOMDRAW	列表视图控件
NMTBCUSTOMDRAW	工具条控件
NMTTCUSTOMDRAW	工具提示控件
NMTVCUSTOMDRAW	树状视图控件

8.1.2 循环绘制、绘制阶段与通告消息

与所有的Microsoft Windows应用程序一样,通用控件基于从系统或其他应用程序所接收到的消息周期性地对自己进行绘制与擦除。控件对自己进行绘制与擦除的过程称为循环绘制 (paint cycle)。那些支持定制绘图特性的控件将在每个绘制循环中周期性地发送NM_CUSTOMDRAW通告消息。伴随这个通告消息一同发送的还有NMCUSTOMDRAW数据结构或者另一个以NMCUSTOMDRAW数据结构作为其第一个数据成员的数据结构。

NMCUSTOMDRAW数据结构中所包含的一个信息就是绘制循环所处的当前阶段,这就是所谓的绘制阶段,绘制阶段用这个数据结构中的dwDrawStage数据成员进行表示。控件将负责向自己的上一层控件或者窗口报告四个基本的控件绘制阶段。这些基本的(也是全局性的)绘制阶段在数据结构中用以下标志位值进行表示(这些值在Commctrl.h头文件中进行了定义):

全局绘制阶段值	描 述
CDDS_POSTERASE	在控件的擦除阶段完成之后
CDDS_POSTPAINT	在控件的绘制阶段完成之后
CDDS_PREERASE	在控件的擦除阶段开始之前
CDDS_PREPAINT	在控件的绘制阶段开始之前

上述每个值都可以与CDDS_ITEM标志位结合使用,用来指定与特定的项目相关联的绘制阶段。为了应用的方便,在Commctrl.h头文件中包含有以下与特定项目相关联的值:

与特定项目相关联的绘制阶段值	描 述
CDDS_ITEMPOSTERASE	在某个项目被擦除之后
CDDS_ITEMPOSTPAINT	在某个项目被绘制之后
CDDS_ITEMPREERASE	在某个项目被查出之前
CDDS_ITEMPREPAINT	在某个项目被绘制之前
CDDS_SUBITEM	在4.71版本中有效。如果正在绘制的是一个子项目,那么标志位将与CDDS_ITEMPRWPAINT或CDDS_ITEMPOSTPAINT组合使用。如果CDRF_NOTIFYITEMDRAW是从CDDS_PREPAINT所返回的,那么这个标志位将被设置

应用程序必须处理NM_CUSTOMDRAW通告消息,然后返回一个特定的值,用来告知控件如何动作。关于返回值的更多信息,请参见后续的章节。

8.1.3 利用定制绘图服务

利用定制绘图功能的关键在于如何响应某个控件所发送的NM_CUSTOMDRAW通告消息。来自应用程序中作为对这些通告消息进行响应的返回值决定了在那个特定绘制阶段控件的行为。

本节将讲述应用程序如何使用NM_CUSTOMDRAW通告消息返回值来决定控件行为的知识。

1. 响应预绘制通告消息

在每个绘制阶段的开始时,控件都将会发送NM_CUSTOMDRAW通告消息,用来指定NMCUSTOMDRAW数据结构中dwDrawStage数据成员的CDDS_PREPAINT值。应用程序返回给

第一个通告消息的值可以说明控件将如何以及何时为后续的绘制阶段发送定制绘制通告消息。作为对第一个通告消息的响应，应用程序可以返回以下标志位值的组合：

返回值	效果
CDRF_DODEFAULT	控件将重新绘制自己。它不会为这个绘制阶段发送附加的NM_CUSTOMDRAW通告消息。这个标志位不能与其他任何标志位一同使用
CDRF_NOTIFYITEMDRAW	控件将向父窗口报告任何与特定项目相关联的绘制操作。在绘制项目的开始之前以及结束之后，控件都将发送NM_CUSTOMDRAW通告消息
CDRF_NOTIFYPOSTPAINT	在整个控件的绘制循环过程完成之后，控件将发送一个NM_CUSTOMDRAW通告消息
CDRF_SKIPDEFAULT	控件将不执行任何绘制动作

2. 请求特定项目的通告消息

如果应用程序向初始的预先定制绘图通告消息返回了CDRF_NOTIFYITEMDRAW值，那么控件将在绘制循环的每个阶段向控件所要绘制的每个项目发送通告消息。这些特定项目的通告消息中都有NMCUSTOMDRAW数据结构dwDrawStage数据成员的CDDS_ITEMPREPAINT值。用户可以在绘制循环过程完成之后向这些特定项目通告消息返回CDRF_NOTIFYPOSTPAINT值，从而要求控件发送另一个通告消息。否则，控件将返回CDRF_DODEFAULT值，并且控件在开始进行下一个项目的绘制之前不会告知父窗口自己正在进行的绘制阶段。

3. 手工绘制项目

如果应用程序已经绘制了整个项目，它将返回CDRF_SKIPDEFAULT。这样能够允许控件跳过那些不需要进行绘制的项目，从而降低了系统开销。需要记住的是，如果返回这个值，那么就意味着控件将不会对项目的任何部分进行绘制。

4. 改变字体与颜色

应用程序可以使用定制绘图功能来改变某个项目的字体。为了实现这个特性，应该选中所需要的HFONT，并将它放置到设备上下文中，这个设备上下文必须由定制绘图通告消息中相关联的NMCUSTOMDRAW数据结构的hdc数据成员所指定。由于所选择的字体可能会与缺省字体的度量不同，因此必须保证在通告消息的返回值中包含有CDRF_NEWFONT位。关于使用这个功能函数的更多信息，请参见8.3节“使用定制绘图”中的示例代码。应用程序中所指定的字体将用来显示那些没有被选中的项目。对于已经选定的项目，定制绘图不允许用户改变字体的属性。

如果需要改变除了列表视图以及树状视图控件之外其他所有支持定制绘图的控件的文本颜色，就必须在设备上下文中设置所需的文本颜色以及背景颜色，并且这些设备上下文必须由带有SetTextColor函数以及SetBkColor函数的定制绘图通告消息数据结构所提供。如果需要修改列表视图或树状视图中的文本颜色，必须在NMLVCUSTOMDRAW或NMTVCUSTOMDRAW数据结构的clrText数据成员以及clrTextBk数据成员中指定所需的颜色值。

8.2 列表视图控件与树状视图控件定制绘图

在本质上，大多数通用控件都是按照相同的方式进行定制绘图处理的。但是，列表视图控

件以及树状视图控件具有某些要求进行特别定制绘图处理的要求。

对于通用控件的5.0版本，如果通过返回CDRF_NEWFONT的方法来改变字体，那么这两个控件将有可能显示被剪辑的文本。这个特性对于保证与通用控件的先前版本的向后兼容性非常必要。在需要改变列表视图或树状视图控件的字体时，如果在向这个控件添加任何项目之前，发送一个带有其wParam数据成员值被设置为5的CCM_SETVERSION消息，那么就能够得到更好的绘图结果。

列表视图控件定制绘图

由于列表视图控件具有子项目并且拥有多个显示方式，因此必须采用与其他通用控件有所不同的方式来对NM_CUSTOMDRAW通告消息进行处理。

对于报告显示方式，应该进行以下处理：

1) 在第一个NM_CUSTOMDRAW通告消息中，NMCUSTOMDRAW数据结构的dwDrawStage数据成员被设置为CDDS_PREPAINT。最后返回的是CDRF_NOTIFYITEMDRAW。

2) 然后，将会接收到一个NM_CUSTOMDRAW通告消息，这个通告消息中的dwDrawStage数据成员被设置为CDDS_ITEMPREPAINT。如果指定了新的字体或颜色，然后返回了CDRF_NEWFONT，那么项目中的所有子项目将被修改。相反，如果需要分别对每个子项目进行处理，那么就应该返回CDRF_NOTIFYSUBITEMDRAW。

3) 如果在上一步中返回值是CDRF_NOTIFYITEMDRAW，那么将会接收到来自每个子项目的NM_CUSTOMDRAW通告消息，这些通告消息中的dwDrawStage数据成员被设置为CDDS_SUBITEM | CDDS_PREPAINT。如果需要为某个子项目改变它的字体或颜色，那么就应该指定新的字体或颜色并返回CDRF_NEWFONT。

对于大图标方式、小图标方式以及列表方式，应该进行以下处理：

1) 在第一个NM_CUSTOMDRAW通告消息中，NMCUSTOMDRAW数据结构的dwDrawStage数据成员被设置为CDDS_PREPAINT。最后返回的是CDRF_NOTIFYITEMDRAW。

2) 然后，将会接收到一个NM_CUSTOMDRAW通告消息，这个通告消息中的dwDrawStage数据成员被设置为CDDS_ITEMPREPAINT。通过指定新字体以及颜色并返回CDRF_NEWFONT，就可以改变某个项目的字体或颜色。由于这些模式没有子项目，因此不会接收到任何其他的NM_CUSTOMDRAW通告消息。

在下一节中将给出一个列表视图的NM_CUSTOMDRAW通告消息处理程序的实例。

8.3 使用定制绘图

以下代码段是WM_NOTIFY处理程序的一部分，它演示了如何对发送给列表视图控件的定制绘图通告消息进行处理：

```
LPNMLISTVIEW pnm = (LPNMLISTVIEW)lParam;

switch (pnm->hdr.code){
    ...
    case NM_CUSTOMDRAW:
```

```

LPNMLVCUSTOMDRAW lplvcd =(LPNMLVCUSTOMDRAW)lParam;

switch(lplvcd->nmcd.dwDrawStage){
case CDDS_PREPAINT :
    return CDRF_NOTIFYITEMDRAW;

case CDDS_ITEMPREPAINT:
    SelectObject(lplvcd->nmcd.hdc,
        GetFontForItem(lplvcd->nmcd.dwItemSpec,
            lplvcd->nmcd.lItemlParam));
    lplvcd->clrText =GetColorForItem(lplvcd->nmcd.dwItemSpec,
        lplvcd->nmcd.lItemlParam);
    lplvcd->clrTextBk =GetBkColorForItem(lplvcd->nmcd.dwItemSpec,
        lplvcd->nmcd.lItemlParam);

/*At this point,you can change the background colors
for the item and any subitems and return CDRF_NEWFONT.
If the list-view control is in report mode,you can
return CDRF_NOTIFYSUBITEMREDRAW to customize the item 's
subitems individually */

    ...
    return CDRF_NEWFONT;
    //or return CDRF_NOTIFYSUBITEMREDRAW;

case CDDS_SUBITEM |CDDS_PREPAINT:
    SelectObject(lplvcd->nmcd.hdc,
        GetFontForSubItem(lplvcd->nmcd.dwItemSpec,
            lplvcd->nmcd.lItemlParam,
            lplvcd->iSubItem));
    lplvcd->clrText =GetColorForSubItem(lplvcd->nmcd.dwItemSpec,
        lplvcd->nmcd.lItemlParam,
        lplvcd->iSubItem));
    lplvcd->clrTextBk =GetBkColorForSubItem(lplvcd->nmcd.dwItemSpec,
        lplvcd->nmcd.lItemlParam,
        lplvcd->iSubItem));

/*This notification is received only if you are in report
mode and returned CDRF_NOTIFYSUBITEMREDRAW in the previous
step.At this point,you can change the background colors
for the subitem and return CDRF_NEWFONT.*/

    ...
    return CDRF_NEWFONT;
}
...
}

```

其中，第一个NM_CUSTOMDRAW通告消息将NMCUSTOMDRAW数据结构中dwDrawStage数据成员的值设置为CDDS_PREPAIN。如果处理程序返回CDRF_NOTIFYITEMDRAW，则表明处理程序希望能够单独地修改一个或多个项目。然后，控件将发送一个其每个项目的dwDrawStage数据成员被设置为CDDS_PREPAIN的NM_CUSTOMDRAW 通告消息。如果处理程序返回CDRF_NOTIFYITEMDRAW，则表明处理程序希望修改项目。

在上述步骤中，如果返回的是CDRF_NOTIFYITEMDRAW，那么下一个NM_CUSTOMDRAW通告消息将把dwDrawStage数据成员的值设置为CDDS_ITEMPREPAIN。这时，处理程序将获取当前颜色与字体值，同时，用户可以为小图标、大图标以及列表方式指定新的颜色与字体值。如果控件当前正处于报告方式，也可以设置新的颜色与字体值，作用于项目的所有子项目。如果对上述情况中所说明的值作了任何修改，那么就应该返回CDRF_NEWFONT。如果控件处于报告方式并且用户希望单独地处理子项目，那么就应该返回CDRF_NOTIFYITEMDRAW。

只有在控件处于报告方式，并且用户在以上步骤中返回了CDRF_NOTIFYSUBITEMDRAW时，最后一个通告消息才被发送。修改字体与颜色的方法与前面的步骤相同，只是这里所做的任何修改仅仅能够作用于单个子项目。如果对颜色或字体作了修改，那么就应该返回CDRF_NEWFONT，以告知控件所进行的修改。

8.4 定制绘图参考

8.4.1 定制绘图通告消息

NM_CUSTOMDRAW

这个通告消息由某些通用控件所发送，用来告知这些通用控件的父窗口当前正在进行的绘制操作。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CUSTOMDRAW
#ifdef LIST_VIEW_CUSTOM_DRAW
    lpNMCustomDraw = (LPNMLVCUSTOMDRAW)lParam;
#elif TOOL_TIPS_CUSTOM_DRAW
    lpNMCustomDraw = (LPNMTCUSTOMDRAW)lParam;
#elif TREE_VIEW_CUSTOM_DRAW
    lpNMCustomDraw = (LPNMTCVCUSTOMDRAW)lParam;
#elif TOOL_BAR_CUSTOM_DRAW
    lpNMCustomDraw = (LPNMTCBCUSTOMDRAW)lParam;
#else
    lpNMCustomDraw = (LPNMCUSTOMDRAW)lParam;
#endif
```

参数

lpNMCustomDraw：与定制绘图相关的数据结构的地址，在这个数据结构中包含有关于绘制操作的信息。下表中列出了能够发送这个通告消息的控件以及相应的数据结构：

控 件	数据结构
列表视图控件	NMLVCUSTOMDRAW
工具条控件	NMTBCUSTOMDRAW
工具提示控件	NMTTCUSTOMDRAW
树状视图控件	NMTVCUSTOMDRAW
所支持的其他所有控件	NMCUSTOMDRAW

返回值

应用程序所能够返回的值取决于当前所处的控件绘制阶段。NMCUSTOMDRAW数据结构中dwDrawStage数据成员保存有一个用来说明当前绘制阶段的值。所返回的值必须是以下值之一：

如果dwDrawStage数据成员的值为CDDS_PREPAINT：

返 回 值	描 述
CDRF_DODEFAULT	控件将绘制自身。并且它不会为这个绘制阶段发送任何其他的NM_CUSTOMDRAW消息
CDRF_NOTIFYITEMDRAW	控件将告知父窗口任何与项目相关的绘制操作。它将在进行项目的绘制之前以及绘制完成之后发送NM_CUSTOMDRAW通告消息
CDRF_NOTIFYPOSTERASE	控件将在擦除某个项目之后告知父窗口
CDRF_NOTIFYPOSTPAINT	控件将在绘制某个项目之后告知父窗口

如果dwDrawStage数据成员的值为CDDS_ITEMPREPAINT：

返 回 值	描 述
CDRF_NEWFONT	应用程序为这个项目指定了新的字体，控件将使用这个新字体。关于改变字体更多的信息，请参见8.1.3节中的“修改字体与颜色”
CDRF_NOTIFYSUBITEMDRAW	对4.71版本有效。在每个列表视图子项目被绘制之前，应用程序将接收到一个NM_CUSTOMDRAW通告消息，在这个通告消息中的dwDrawStage数据成员被设置为CDDS_ITEMPREPAINT CDDS_SUBITEM。然后，就可以为每个子项目单独地指定字体与颜色，或者返回CDRF_DODEFAULT，表示需要进行缺省处理
CDRF_SKIPDEFAULT	应用程序自行绘制项目，控件不再对项目进行绘制

说明

目前，以下控件能够支持定制绘图功能：头标控件、列表视图控件、Rebar控件、工具条控件、工具提示控件、轨迹条控件以及树状视图控件。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

8.3节。

8.4.2 定制绘图数据结构

NMCUSTOMDRAW

在这个数据结构中包含NM_CUSTOMDRAW通告消息中所特有的信息。

```
typedef struct tagNMCUSTOMDRAWINFO {
    NMHDR    hdr;
    DWORD    dwDrawStage;
    HDC      hdc;
    RECT      rc;
    DWORD    dwItemSpec;
    UINT     uItemState;
    LPARAM   lItemParam;
} NMCUSTOMDRAW, FAR *LPNMCUSTOMDRAW;
```

成员

hdr：包含有关于本通告消息的特定信息的NMHDR数据结构。

dwDrawStage：当前绘制阶段。这个值可以是以下值之一：

全局值	描述
CDDS_POSTERASE	在擦除阶段完成之后
CDDS_POSTPAINT	在绘制阶段完成之后
CDDS_PREERASE	在擦除阶段开始之前
CDDS_PREPAINT	在绘制阶段开始之前
项目特定值	描述
CDDS_ITEM	表示dwItemSpec、uItemState以及lItemParam数据成员有效
CDDS_ITEMPOSTERASE	在一个项目被擦除之后
CDDS_ITEMPOSTPAINT	在一个项目被绘制之后
CDDS_ITEMPREERASE	在一个项目被擦除之前
CDDS_ITEMPREPAINT	在一个项目被绘制之前
CDDS_SUBITEM	在4.71版本中有效。如果正在绘制某个子项目，那么这个值就是与CDDS_ITEMPREPAINT或CDDS_ITEMPOSTPAINT组合使用的标志位值。只有当从CDDS_PREPAINT中返回了CDRF_NOTIFYITEMDRAW值时，这个值才被设置

hdc：控件设备上下文的句柄，可以用这个HDC来执行任何GDI功能。

rc：RECT数据结构，这个数据结构用来描述正在被绘制区域的边界矩形。这个数据成员只能被CDDS_ITEMPREPAINT通告消息初始化。

dwItemSpec：项目编号，包含在这个数据成员中的值取决于正在发送通告消息的控件的类

型。如果需要了解包含在这个数据成员中的特定控件，请参见NM_CUSTOMDRAW通告消息参考。

ulItemState: 当前项目状态，这个值是以下值的组合：

值	描 述
CDIS_CHECKED	项目被选中
CDIS_DEFAULT	项目目前处于缺省状态
CDIS_DISABLED	项目功能被关闭
CDIS_FOCUS	项目当前处于焦点状态
CDIS_GRAYED	项目当前处于灰色显示状态
CDIS_HOT	项目当前处于指针状态（“热点”）
CDIS_INDETERMINATE	项目当前处于中间状态
CDIS_MARKED	项目当前被打上对号标志位，其具体含义取决于具体实现
CDIS_SELECTED	项目当前被选中

lItemlParam: 由应用程序所定义的项目数据。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。



第9章 动画控件

9.1 动画控件概述

动画控件(animation control)是一个显示AVI剪辑的窗口。AVI剪辑由一系列如同电影帧图像的位图帧组成,动画控件只能显示没有包含音频信息的AVI剪辑。

9.1.1 关于动画控件

动画控件的常见应用就是在一个耗时较长的操作中表明系统正在进行的活动。之所以能够实现这个功能,是因为在显示AVI剪辑的同时操作的线程能够继续执行。例如,当系统正在搜索某个文件时,Microsoft Windows 资源管理器中的Find对话框将显示一个正在移动的放大镜。

无论AVI剪辑是来自一个没有经过压缩的AVI文件还是来自一个被采用BI_RLE8编码方式进行压缩的AVI文件,动画控件都可以对这些AVI文件的AVI剪辑进行播放。可以向应用程序中添加AVI剪辑,使其作为AVI资源。也可以将AVI剪辑作为一个单独的AVI文件与应用程序共存。

注意 AVI文件(资源)并不需要有任何声音通道。动画控件所能提供的功能是十分有限的,并且它所提供的功能还处于变化过程中。如果用户需要一个能够为应用程序提供多媒体播放以及录音功能的控件,就应该考虑使用MCIWnd控件。

1. 创建动画控件

动画控件属于ANIMATE_CLASS窗口类,用户可以使用CreateWindow函数或Animate_Create宏来创建动画控件。如果使用的是宏来创建动画控件,那么这个宏将把动画控件定位于父窗口的左上方,并且如果没有指定ACS_CENTER样式,动画控件将基于AVI剪辑中帧的大小来设置控件的宽度与高度。如果指定了ACS_CENTER样式,Animate_Create将把控件的宽度与高度设置为0。用户可以使用SetWindowPos函数来设置控件的位置与大小。

如果在一个对话框中或者从一个对话框资源中创建动画控件,那么当用户关闭对话框时控件将自动被析构。如果在窗口中创建了动画控件,那么就必须显式地对动画控件进行析构。

2. 关于动画控件消息

应用程序通过向动画控件发送消息的方法来打开、播放、停止与关闭相应的AVI剪辑。事实上,不必显式地进行消息的发送,每个消息都有一个或多个宏,能够完成消息的发送动作。

在创建动画控件之后,应用程序将发送ACM_OPEN消息,用来打开一个AVI剪辑并且将这个AVI剪辑加载到内存中。所发送的消息可以指定AVI文件的路径,或者指定AVI资源的资源名。系统将从创建了动画控件的模块中加载AVI资源。

如果动画控件的样式为ACS_AUTOPLAY,那么控件将在AVI文件或AVI资源打开之后立刻播放AVI剪辑。否则,应用程序将使用ACM_PLAY消息来开始进行AVI剪辑的播放。此外,应用

程序还可以发送ACM_STOP消息,从而在任何时候停止AVI剪辑的播放。当控件完成对AVI剪辑的播放或者应用程序向动画控件发送了ACM_STOP消息时,所播放的最后一个帧将仍然在显示器上显示。

动画控件可以向自己的父窗口发送两个通告消息:ACN_START通告消息与ACM_STOP通告消息。大多数应用程序将不会对这两个通告消息作任何处理。

如果需要关闭AVI文件或AVI资源,并且将它从内存中删除,应用程序可以使用Animate_Close宏来完成这个动作,Animate_Close宏将会发送一个AVI文件名或AVI资源名被设置为NULL的ACM_OPEN消息。

缺省消息处理

本节将讨论窗口过程为ANIMATE_CLASS窗口类所处理的窗口消息。

消 息	所执行的处理动作
WM_CLOSE	释放与动画控件相关联的AVI文件或AVI资源
WM_DESTROY	释放AVI文件或AVI资源,释放内部数据结构,然后调用DefWindowProc函数
WM_ERASEBKGDND	利用静态控件的当前背景色擦除窗口的背景色
WM_NCCREATE	分配并初始化内部数据结构,然后调用DefWindowProc函数
WM_NCHITTEST	返回HTTRANSPARENT鼠标击中测试值
WM_PAINT	在动画控件中绘制AVI帧
WM_SIZE	检查控件是否具有ACS_CENTER样式。如果控件没有此样式,将调用DefWindowProc函数。否则,将把动画显示在控件的中央位置,将控件设置为无效,然后调用DefWindowProc函数

9.1.2 使用动画控件

本节将给出一个演示如何创建动画控件并且在动画控件中显示AVI剪辑的例子。

1. 创建动画控件

以下函数将在一个对话框中创建动画控件。所创建的动画控件位于指定控件的下方,并且这个动画控件的大小是基于AVI剪辑中帧的大小来确定的。

```
//CreateAnimationCtrl -creates an animation control,
// positions it below the specified control in a
// dialog box,and opens the AVI clip for the
// animation control.
//Returns the handle to the animation control.
//hwndDlg -handle to the dialog box.
//nIDctl -identifier of the control below which the
// animation control is to be positioned.
//
//Constants
// IDC_ANIMATE -identifier of the animation control.
// CX_FRAME,CY_FRAME -width and height of the frames
// in the AVI clip.
HWND CreateAnimationCtrl(HWND hwndDlg,int nIDctl)
```

```

{
    HWND hwndAnim = NULL;
    RECT rc;
    POINT pt;

    //Create the animation control.
    hwndAnim =Animate_Create(hwndDlg, IDC_ANIMATE,
        WS_BORDER |WS_CHILD, g_hinst);

    //Get the screen coordinates of the specified control
    //button.
    GetWindowRect(GetDlgItem(hwndDlg, nIDCtl), &rc);

    //Convert the coordinates of the lower-left corner to
    //client coordinates.
    pt.x =rc.left;
    pt.y =rc.bottom;
    ScreenToClient(hwndDlg, &pt);

    //Position the animation control below the Stop button.
    SetWindowPos(hwndAnim, 0, pt.x, pt.y +20,
        CX_FRAME, CY_FRAME,
        SWP_NOZORDER |SWP_DRAWFRAME);
    //Open the AVI clip, and show the animation control.
    Animate_Open(hwndAnim, 'SEARCH.AVI');
    ShowWindow(hwndAnim, SW_SHOW);

    return hwndAnim;
}

```

2. 控制AVI剪辑

下列函数将使用动画控件宏来控制AVI剪辑在动画控件中的显示。

```

//DoAnimation -plays, stops, or closes an animation
// control's AVI clip, depending on the value of an
// action flag.
//hwndAnim -handle to an animation contro
//nAction -flag that determines whether to play, stop,
// or close the AVI clip.
void DoAnimation(HWND hwndAnim, int nAction)
{
    switch (nAction){
        case PLAYIT:

            //Play the clip continuously starting with the
            //first frame.
            Animate_Play(hwndAnim, 0, -1, -1);
            break;

```



```

case STOPIT:
    Animate_Stop(hwndAnim);
    break;

case CLOSEIT:
    Animate_Close(hwndAnim);
    break;

default:
    break;
}
return;
}

```

9.2 动画控件样式

在动画控件中可以使用以下窗口样式:

ACS_AUTOPLAY

只要AVI剪辑被打开,就立刻进行动画的播放。

ACS_CENTER

将动画定位于动画控件窗口中央位置。

ACS_TIMER

缺省情况下,控件将创建一个线程来播放AVI剪辑。如果用户设置了这个标记,那么控件将不创建线程而直接重放AVI剪辑。在系统内部,控件将使用Win32定时器来对动画的重放进行同步。

ACS_TRANSPARENT

允许用户将动画的背景色与动画播放所在窗口的背景色相匹配,从而创建“透明”的背景色。这时,控件将向自己的父窗口发送一个WM_CTLCOLORSTATIC消息。可以使用SetBkColor将设备上下文的背景色设置为合适的值。控件将把动画第一帧中左上方的像素解释为动画的缺省背景色。动画控件在响应WM_CTLCOLORSTATIC消息时,将把所有与这个颜色相同的像素进行重画,从而将这些像素的颜色值设置为所指定的值。

9.3 动画控件参考

9.3.1 动画控件消息

ACM_OPEN

打开一个AVI剪辑,并且将这个AVI剪辑中的第一帧在动画控件中显示出来。用户可以显式地发送这个消息,也可以使用Animate_Open宏或Animate_OpenEx宏来发送这个消息。

```

ACM_OPEN
#if (_WIN32_IE >= 0x0400)

```

```

wParam = (WPARAM) (HINSTANCE) hinst;
#else
wParam = 0;
#endif
lParam = (LPARAM) (LPSTR) lpszName;

```

参数

hinst: 在4.71版本中有效。将要从中加载AVI资源的模块的实例句柄。如果需要让控件使用用来创建窗口的HINSTANCE值，那么就应该将这个参数值设置为NULL。需要说明的是，如果这个窗口是被一个DLL所创建的，那么hinst参数的缺省值应该是这个DLL的HINSTANCE值，而不应该是调用这个DLL的应用程序的HINSTANCE值。

lpszName: 一个缓冲器的地址，在这个缓冲器中包含有AVI文件的路径或AVI资源的资源名。此外，这个参数还可以在低字节部分包含AVI资源的标识符，而在高字节部分为0。如果需要创建这个值，就应该使用MAKEINTRESOURCE宏。动画控件将从由实例句柄所指定的模块中加载AVI资源，而这个实例句柄必须是传递给创建本动画控件的CreateWindow函数、Animate_Create宏或对话框创建函数的那个实例句柄。在4.71版本及其后续版本中，AVI资源是从由hinst所指定的模块中进行加载的，AVI资源必须具有“AVI”类型。

如果这个参数为NULL，系统将关闭先前为这个特定的动画控件所打开的AVI文件（若存在）。

返回值

如果操作成功，则返回非零值；否则，返回0。

说明

由lpszName参数所指定的AVI文件或AVI资源必须没有包含任何音频信息。

对于Windows 95系统，动画控件仅仅能够对以ANSI字符串作为lpszName参数值的ANSI版本的消息（ACM_OPENA）进行响应，而对于Unicode版本的ACM_OPENW消息，系统将不能作任何响应。

只能打开没有音频信息的AVI剪辑，如果lpszSource参数指定了一个带有声音的AVI剪辑，ACM_OPEN与Animate_Open将会失败。

可以使用Animate_Close来关闭先前为某个指定的动画控件所打开的AVI文件或AVI资源。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

ACM_PLAY

在动画控件中播放AVI剪辑。控件能够在相同的背景下而且线程继续执行的情况下播放AVI剪辑。用户可以显式地发送这个消息，也可以使用Animate_Play宏来发送这个消息。

ACM_PLAY

```
wParam = (WPARAM)(UINT) cRepeat;
lParam = (LPARAM) MAKELONG(wFrom, wTo);
```

参数

cRepeat: 反复播放AVI剪辑的次数。如果这个参数的值为-1, 那么AVI剪辑将无限次地进行播放。

wFrom: 一个基于0的索引值, 用来表明动画开始播放的帧的起始位置, 并且这个参数值必须小于65 536。如果参数值为0, 那么就表示必须从AVI剪辑的第一帧开始进行播放。

wTo: 一个基于0的索引值, 用来表明动画停止播放的帧的位置, 并且这个参数值必须小于65 536。如果这个参数值为-1, 那么就表示必须播放到AVI剪辑的结尾处。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

说明

可以使用Animate_Seek来使得动画控件显示AVI剪辑中的某个特定帧。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

ACM_STOP

在动画控件中停止播放AVI剪辑。可以显式地发送这个消息, 也可以使用Animate_Stop宏来发送这个消息。

```
ACM_STOP
wParam = 0;
lParam = 0;
```

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

9.3.2 动画控件宏

Animate_Close

关闭一个AVI剪辑, 并且在动画控件中显示这个AVI剪辑的第一帧。可以使用这个宏或者发

送ACM_OPEN消息来完成此动作。

```
BOOL Animate_Close(  
    HWND hwnd  
);
```

参数

hwnd: 动画控件的句柄。

返回值

总是返回FALSE。

说明

可以使用Animate_Close来关闭先前为某个特定的动画控件所打开的AVI文件或AVI资源。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Animate_Open、MAKEINTRESOURCE。

Animate_Create

创建一个动画控件。Animate_Create将调用CreateWindow函数来创建动画控件。

```
HWND Animate_Create(  
    HWND hwndP ,  
    UINT id ,  
    DWORD dwStyle ,  
    HINSTANCE hInstance  
);
```

参数

hwndP: 父窗口的句柄。

id: 动画控件子窗口的标识符。

dwStyle: 窗口样式。关于动画控件样式值的列表, 请参见9.2节。

hInstance: 创建动画控件的实例模块的句柄。

返回值

返回动画控件的句柄。

说明

如果指定了ACS_CENTER样式, 那么Animate_Create宏将把动画控件的宽度与高度设置为0。

如果没有指定ACS_CENTER样式, 那么Animate_Create将基于AVI剪辑中帧的大小来设置动画控件的宽度与高度。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Animate_Open

打开一个AVI剪辑, 并且在动画控件中显示这个AVI剪辑的第一帧。可以使用这个宏或者显式发送ACM_OPEN消息来完成此动作。

```
BOOL Animate_Open(
    HWND hwnd,
    LPSTR lpszName
);
```

参数

hwnd: 动画控件的句柄。

lpszName: 一个缓冲器的地址, 在这个缓冲器中包含有AVI文件的路径或AVI资源的资源名。此外, 这个参数还可以在低字节部分包含AVI资源的标识符, 而在高字节部分为0。如果需要创建这个值, 就应该使用MAKEINTRESOURCE宏。动画控件将从由实例句柄所指定的模块中加载AVI资源, 而这个实例句柄必须是传递给创建本动画控件的CreateWindow函数、Animate_Create宏或对话框创建函数的那个实例句柄。

由lpszName所指定的AVI文件或AVI资源必须不包含任何音频信息。

如果这个参数为NULL, 系统将关闭先前为这个特定的动画控件所打开的AVI文件(若存在)。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

说明

只能打开没有声音的AVI剪辑。如果lpszSource参数中指定了一个带有声音信息的AVI剪辑, 那么ACM_OPEN与Animate_Open将会失败。

可以利用Animate_Close来关闭先前在特定的动画控件中所打开的AVI文件或AVI资源。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Animate_OpenEx

从一个特定的模块中打开AVI剪辑资源, 并且在动画控件中显示这个AVI剪辑的第一帧。可以使用这个宏或者显式地发送ACM_OPEN消息来完成这个动作。

```

BOOL Animate_OpenEx(
    HWND hwnd ,
    HINSTANCE hinst ,
    LPSTR lpszName
);

```

参数

hwnd: 动画控件句柄。

hinst: 将要从中加载AVI资源的模块的实例句柄。如果这个参数值为NULL, 那么将从创建这个动画控件的模块中加载AVI资源。

lpszName: 一个缓冲器的地址, 在这个缓冲器中包含有AVI文件的路径或AVI资源的资源名。此外, 这个参数还可以在低字节部分包含AVI资源的标识符, 而在高字节部分为0。如果需要创建这个值, 就应该使用MAKEINTRESOURCE宏。动画控件将从由hinst参数中所指定的模块中加载AVI资源。

由lpszName所指定的AVI文件或AVI资源必须不包含任何音频信息。

如果这个参数为NULL, 系统将关闭先前为这个特定的动画控件所打开的AVI文件(若存在)。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

说明

只能打开没有声音的AVI剪辑。如果lpszSource参数中指定了一个带有声音信息的AVI剪辑, 那么ACM_OPEN与Animate_Open将会失败。

可以利用Animate_Close来关闭先前在特定的动画控件中所打开的AVI文件或AVI资源。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Animate_Play

在动画控件中播放AVI剪辑。动画控件将在线程继续执行的情况下在背景上播放这个AVI剪辑。可以使用这个宏或显式地发送ACM_PLAY消息来完成此动作。

```

BOOL Animate_Play(
    HWND hwndAnim ,
    UINT wFrom ,
    UINT wTo ,

```



```
UINT cRepeat
);
```

参数

hwndAnim: 将要在其中播放AVI剪辑的动画控件的句柄。

wFrom: 一个基于0的索引值, 用来标明动画开始进行播放的位置。这个参数值必须小于65 536。如果参数值为0, 那么就表示应该从AVI剪辑的第一帧开始播放。

wTo: 一个基于0的索引值, 用来标明动画结束播放的位置。这个参数值必须小于65 536, 如果参数的值为-1, 就表示AVI剪辑将播放到最后的帧。

cRepeat: 反复进行AVI剪辑播放的次数, 如果这个参数的值为-1, 就表示进行无限次的播放。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

说明

可以使用Animate_Sseek来引导动画控件显示AVI剪辑中的特定帧。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Animate_Sseek

用来引导动画控件显示AVI剪辑中特定的帧。动画控件能够在线程继续执行的条件下在背景中显示AVI剪辑。可以使用这个宏或者显式地发送ACM_PLAY消息来完成此动作。

```
BOOL Animate_Sseek(
    HWND hwndAnim,
    UINT wFrame
);
```

参数

hwndAnim: 将要在其中显示AVI帧的动画控件句柄。

wFrame: 将要显示的帧基于0的索引。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Animate_Play。

Animate_Stop

停止在动画控件中播放AVI剪辑。可以使用这个宏或者显式地发送ACM_STOP消息来完成此动作。

```
BOOL Animate_Stop(  
    HWND hwnd  
);
```

参数

hwnd: 动画控件的句柄。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

9.3.3 动画控件通告消息

ACN_START

这个通告消息用来告知动画控件的父窗口, 相对应的AVI剪辑已经开始进行了播放。这个通告消息是以WM_COMMAND消息的形式进行发送的。

ACN_START

返回值

所得到的返回值将被忽略。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

ACN_STOP

这个通告消息用来告知动画控件的父窗口, 相对应的AVI剪辑已经停止了播放。这个通告消息是以WM_COMMAND消息形式进行发送的。

ACN_STOP

返回值

所得到的返回值将被忽略。

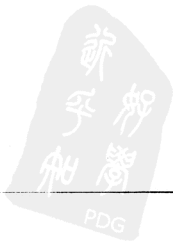
环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。



第10章 扩展组合框控件

扩展组合框控件 (ComboBoxEx control) 就是组合框控件的扩展, 能支持项目图像。为了使得项目图像能够非常容易被访问, 扩展组合框控件提供了图像列表支持功能。使用扩展组合框控件, 可以在不必手工地对项目的图像进行绘制的情况下就能够提供组合框所具有的所有功能。

10.1 关于扩展组合框控件

从本质上来说, 扩展组合框控件将创建一个子组合框并基于所分配的图像列表执行组合框绘制任务。因此, 在扩展组合框控件中, 所采用的是CBS_OWNERDRAWFIXED样式, 并且在创建这个控件时不必指定这个样式。由于图像列表可用来提供项目图像, 因此不能使用CBS_OWNERDRAWVARIABLE样式。

必须调用InitCommonControlsEx函数并且在相应的INITCOMMONCONTROLSEX数据结构中指定ICC_USEREX_CLASSES值, 进行扩展组合框控件的初始化。

可以使用CreateWindowEx函数并且将WC_COMBOBOXEX指定为一个窗口类来创建扩展组合框控件。这个所指定的窗口类必须上面所说明的InitCommonControlsEx函数调用时进行注册。

在创建扩展组合框控件时, 并没有任何缺省图像列表。如果需要项目图像, 就必须首先为扩展组合框控件创建一个图像列表并且将这个图像列表分配给使用CBEM_SETIMAGELIST消息的控件。如果没有为扩展组合框控件分配任何图像列表, 那么控件将仅仅显示项目的文本。

10.1.1 扩展组合框控件样式

扩展组合框控件只能支持以下组合框样式:

- CBS_SIMPLE
- CBS_DROPDOWN
- CBS_DROPDOWNLIST
- WS_CHILD

此外, 还有一些仅仅对于扩展组合框控件有效的扩展样式。

注意 在某些情况下, CBS_SIMPLE样式可能不能正确地工作。

由于扩展组合框控件能够基于一个已经分配的图像列表来执行所有者绘制任务, 因此扩展组合框控件隐含地将使用CBS_OWNERDRAWFIXED样式; 在创建扩展组合框控件时, 并不需要指定这个样式。由于图像列表可以用于提供关于项目的图像, 因此不能使用CBS_OWNERDRAWVARIABLE样式。此外, 扩展组合框控件还支持一些扩展的样式, 能提供更多的特性。

10.1.2 扩展组合框控件项目

扩展组合框控件使用COMBOBOXEXITEM数据结构来维护项目信息。在这个数据结构中,包含有关于项目索引、图像索引(普通、选择状态以及覆盖)、缩排值、文本字符串以及特定项目值在内的数据成员。

通过使用消息,扩展组合框控件能够提供对项目非常简单的访问与处理。如果需要添加或删除某个项目,就应该发送CBEM_INSERTITEM或CBEM_DELETEITEM消息。利用CBEM_SETITEM消息,可以修改控件中的当前项目。

10.1.3 信号回叫项目

扩展组合框控件支持信号回叫项目属性。利用CBEM_INSERTITEM,可以在将某个项目添加到扩展组合框控件中时把这个项目指定为信号回叫项目。当向一个项目的COMBOBOXEXITEM数据结构分配值时,必须指定适当的信号回叫标记值。下表列出了COMBOBOXEXITEM数据结构的成员以及相对应的信号回叫标记值:

数据成员	信号回叫值
pszText	LPSTR_TEXTCALLBACK
iImage	I_IMAGECALLBACK
iSelectedImage	I_IMAGECALLBACK
iOverlay	I_IMAGECALLBACK
iIndent	I_INDENTCALLBACK

扩展组合框控件通过发送CBEN_GETDISPINFO通告消息方法来获取关于信号回叫项目的信息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。当应用程序处理这个消息时,它就必须为扩展组合框控件提供所请求的信息。如果将COMBOBOXEXITEM数据结构中mask数据成员的值设置为CBEIF_DI_SETITEM,扩展组合框控件将存储这个项目的数据,并且在今后不请求此数据。

10.1.4 扩展组合框控件图像列表

如果希望某个扩展组合框控件显示带有图标的项目,就必须提供图像列表。扩展组合框控件最多能够为一个项目支持三个图像:其中一个用于项目的选中状态,一个用于项目的非选中状态,最后一个用于覆盖图状态。可以使用CBEM_SETIMAGELIST消息将一个现有的图像列表分配给扩展组合框控件。

在COMBOBOXEXITEM数据结构中包含有用来为每个图像列表(选中图像、非选中图像以及覆盖图像)中的图像提供索引的数据成员。对于每个项目,应该将这些数据成员设置为显示相应的图像。用户并不需要为每个类型的图像指定图像索引,而是可以按照自己的需要进行图像类型的分配,但应该总是保证将COMBOBOXEXITEM数据结构中的mask数据成员设置为表示正在被使用的那个数据成员。扩展组合框控件将忽略那些没有被标记为合法的数据成员。

注意 如果使用的是CBS_SIMPLE样式,那么图标将不会被显示。

10.1.5 关于扩展组合框控件的通告消息

扩展组合框控件通过发送通告消息方法来报告发生在自身的改变以及请求信号回叫项目信息。扩展组合框控件的父窗口能够接收来自包含在这个扩展组合框控件中组合框的所有 WM_COMMAND 消息。扩展组合框控件使用 WM_NOTIFY 消息来发送自己特定的通告消息。因此，扩展组合框控件的所有者必须能够进行这两种形式的通告消息的处理。

下表列出了扩展组合框控件所特有的通告消息：

通告消息	描 述
CBEN_BEGINEDIT	用来报告用户已经激活了下拉列表或者单击了控件的编辑框
CBEN_DELETEITEM	用来报告有一个项目被删除
CBEN_ENDEDIT	用来报告用户已经从下拉列表中选定了一个项目或者完成了编辑框中的编辑操作
CBEN_GETDISPINFO	用来请求获取关于项目属性的消息
CBEN_INSERTITEM	用来报告有一个项目已经被插入到控件中

10.1.6 扩展组合框控件消息的转发

下面列出了扩展组合框控件将转发到子组合框中的标准组合框消息，其中某些消息可能会在被扩展组合框控件转发之前或者转发之后被扩展组合框控件所处理：

- CB_DELETESTRING
- CB_FINDSTRINGEXACT
- CB_GETCOUNT
- CB_GETCURSEL
- CB_GETDROPPEDCONTROLRECT
- CB_GETDROPPEDSTATE
- CB_GETEXTENDEDUI
- CB_GETITEMDATA
- CB_GETITEMHEIGHT
- CB_GETLBTEXT
- CB_GETLBTEXTLEN
- CB_LIMITTEXT
- CB_RESETCONTENT
- CB_SETCURSEL
- CB_SETDROPPEDWIDTH
- CB_SETEXTENDEDUI
- CB_SETITEMDATA
- CB_SETITEMHEIGHT
- CB_SHOWDROPDOWN

下面列出了扩展组合框控件将转发给父窗口的窗口消息：

- WM_COMMAND (这里, 包括了所有的CBN_通告消息)
- WM_NOTIFY

10.2 使用扩展组合框控件

10.2.1 创建扩展组合框控件

如果需要创建扩展组合框控件, 就应该使用WC_COMBOBOXEX作为窗口类, 调用CreateWindowEx函数。在此之前, 首先必须通过调用InitCommonControlsEx函数进行窗口类的注册, 在调用这个函数时, 应该为相应的INITCOMMONCONTROLSEX数据结构指定ICC_USEREX_CLASSES位值。

下面的应用程序和函数将在主窗口中创建一个带有CBS_DROPDOWN样式的扩展组合框控件:

```
//CreateComboEx -Registers the ComboBoxEx window class
//and creates a ComboBoxEx control in the client area of
//the main window.
//
//g_hwndMain -A handle to the main window.
//g_hinst    -A handle to the program instance.

HWND WINAPI CreateComboEx(void)
{
    HWND hwnd;
    INITCOMMONCONTROLSEX icex;

    icex.dwSize = sizeof(INITCOMMONCONTROLSEX);
    icex.dwICC = ICC_USEREX_CLASSES;

    InitCommonControlsEx(&icex);

    hwnd =CreateWindowEx(0,WC_COMBOBOXEX,NULL,
                        WS_BORDER |WS_VISIBLE |
                        WS_CHILD |CBS_DROPDOWN,
                        //No size yet -resize
                        //after setting image list.
                        0,        //Vertical position of Combobox
                        0,        //Horizontal position of Combobox
                        0,        //Sets the width of Combobox
                        100,     //Sets the height of Combobox
                        g_hwndMain,
                        NULL,
                        g_hinst,
                        NULL);

    return (hwnd);
}
```

10.2.2 准备扩展组合框项目与图像

在将一个项目添加到扩展组合框控件之前，首先必须定义COMBOBOXEXITEM数据结构。在定义的过程中，应该将这个数据结构的mask数据成员设置为希望控件使用的数据成员。将数据结构中指定的数据成员设置为所需的值，然后发送CBEM_INSERTITEM消息，从而将项目添加到控件中。

下面应用程序所定义的函数能够将15个项目添加到一个现有的扩展组合框控件中。需要说明的是，COMBOBOXEXITEM数据结构中的mask数据成员中包含有一些标记值，这些标记值可以用来告知控件为每个项目应该显示的图像：

```
#define MAX_ITEMS 15

//AddItems -Uses the CBEM_INSERTITEM message to add items
//to an existing ComboBoxEx control.

BOOL WINAPI AddItems(HWND hwndCB)
{
    //Declare and init locals.
    COMBOBOXEXITEM cbei;
    int iCnt;

    typedef struct {
        int iImage;
        int iSelectedImage;
        int iIndent;
        LPCTSTR pszText;
    }ITEMINFO,*PITEMINFO;

    ITEMINFO IInf [ ] ={
        {0,3,0,"first"},
        {1,4,1,"second"},
        {2,5,2,"third"},
        {0,3,0,"fourth"},
        {1,4,1,"fifth"},
        {2,5,2,"sixth"},
        {0,3,0,"seventh"},
        {1,4,1,"eighth"},
        {2,5,2,"ninth"},
        {0,3,0,"tenth"},
        {1,4,1,"eleventh"},
        {2,5,2,"twelfth"},
        {0,3,0,"thirteenth"},
        {1,4,1,"fourteenth"},
        {2,5,2,"fifteenth"}
    };

    //Set the mask common to all items.
```



```

cbei.mask =CBEIF_TEXT |CBEIF_INDENT |
                CBEIF_IMAGE|CBEIF_SELECTEDIMAGE;
for(iCnt=0;iCnt<MAX_ITEMS;iCnt++){
    //Initialize the COMBOBOXEXITEM struct.
    cbei.iItem          =iCnt;
    cbei.pszText         =IInf [iCnt ].pszText;
    cbei.cchTextMax      =sizeof(IInf [iCnt ].pszText);
    cbei.iImage          =IInf [iCnt ].iImage;
    cbei.iSelectedImage  =IInf [iCnt ].iSelectedImage;
    cbei.iIndent         =IInf [iCnt ].iIndent;

    //Tell the ComboBoxEx to add the item.Return
    //FALSE if this fails.
    if(SendMessage(hwndCB,CBEM_INSERTITEM,0,(LPARAM)&cbei)==-1)
        return FALSE;
}
//Assign the existing image list to the ComboBoxEx
//control and return TRUE.
SendMessage(hwndCB,CBEM_SETIMAGELIST,0,(LPARAM)g_himl);

//Set size of control to make sure it's displayed
//correctly now that the image list is set.
SetWindowPos(hwndCB,NULL,20,20,250,120,SWP_NOACTIVATE);

return TRUE;
}

```

10.2.3 支持信号回叫项目

如果应用程序将要在扩展组合框控件中使用信号回叫项目，那么这个应用程序就必须能够处理CBEN_GETDISPINFO通告消息。当扩展组合框控件需要控件所有者提供特定的项目信息时，它将发送此通告消息。关于信号回叫项目的更多信息，请参见10.1.3节。

下面应用程序所定义的函数能够通过为给定的项目提供属性方法来处理CBEN_GETDISPINFO通告消息。需要说明的是，这个函数将把输入COMBOBOXEXITEM数据结构的mask数据成员设置为CBEIF_DI_SETITEM。将mask数据成员设置为这个值能够使得控件保持项目信息，从而不必反复请求此项目的信息：

```

//DoItemCallback -Processes CBEN_GETDISPINFO by
//providing item attributes for a given callback item.

void WINAPI DoItemCallback(PNMCOMBOBOXEX pNMCBex)
{
    DWORD dwMask =pNMCBex->ceItem.mask;

    if(dwMask &CBEIF_TEXT)
        //Provide item text.

```

```

if(dwMask & CBEIF_IMAGE)
    ;//Provide an item image index.

/*
*Provide other callback information as desired.
*/

//Make the ComboBoxEx control hold onto the item
//information.
pNMCBex->ceItem.mask =CBEIF_DI_SETITEM;
}

```

10.2.4 处理扩展组合框通告消息

扩展组合框控件能够通过发送WM_NOTIFY消息方法来告知父窗口所发生的事件，由于扩展组合框控件使用了一个子组合框，因此它将会把自己所接收到的所有WM_COMMAND通告消息转发给父窗口进行处理。这样，应用程序就必须能够处理来自扩展组合框控件的WM_NOTIFY消息并且能够处理来自扩展组合框控件子组合框控件的转发WM_COMMAND消息。

本节中所给的这个代码实例能够通过调用相应的应用程序函数的方法完成来自扩展组合框控件的WM_NOTIFY消息与WM_COMMAND消息。

```

LRESULT CALLBACK WndProc (HWND hwnd,UINT msg,WPARAM wParam,LPARAM lParam)
{
    switch(msg){
        case WM_COMMAND:
            //If this is an 'old style'combo box
            //notification,handle it.
            if((HWND)lParam ==g_hwndCB)
                DoOldNotify(hwnd,wParam);
            break;

        case WM_NOTIFY:
            return (DoCBEXNotify(hwnd,lParam));

        case WM_PAINT:
            hdc =BeginPaint(hwnd,&ps);
            EndPaint(hwnd,&ps);
            break;

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        default:
            return DefWindowProc(hwnd,msg,wParam,lParam);
            break;
    }
}

```

```

    }
    return FALSE;
}

```

10.3 扩展组合框控件的扩展样式

扩展组合框控件能够支持大多数标准的组合框控件样式。此外，扩展组合框控件还能够支持以下的扩展样式，用户可以使用CBEM_SETEXTENDEDSTYLE消息与CBEM_GETEXTENDEDSTYLE消息来设置并获取这些样式：

CBES_EX_CASESENSITIVE：在这个列表中所进行的字符串查找将具有大小写敏感的特性。这些字符串查找包括对编辑框中被输入文本的查找以及对CB_FINDSTRINGEXACT消息中文本的查找。

CBES_EX_NOEDITIMAGE：编辑框以及下拉列表将不会显示项目图像。

CBES_EX_NOEDITIMAGEINDENT：编辑框以及下拉列表将不会显示项目图像。

CBES_EX_NOSIZELIMIT：允许扩展组合框控件在竖直方向上改变自己的大小，并且可以小于包含在扩展组合框控件中的组合方式控件大小。如果扩展组合框控件的大小比内部组合框的大小更小，那么组合框将被截取。

CBES_EX_PATHWORDBREAKPROC：仅仅对于Windows NT有效。编辑框将使用斜线(/)、反斜杠(\)以及小圆点(.)字符作为单词的分隔符。这样能够使得快捷方式以及在路径名以及URL的单词之间的光标移动(CTRL+ARROW)变得十分容易。

注意 如果用户试图对利用CBS_SIMPLE样式所创建的扩展组合框控件设置扩展样式，那么所设置样式可能不能正常工作。同时，CBS_SIMPLE样式也不能与CBES_EX_PATHWORDBREAKPROC扩展样式在一起正常工作。

10.4 扩展组合框控件参考

10.4.1 扩展组合框控件消息

CBEM_DELETEITEM

从扩展组合框控件中删除某个项目。

```

CBEM_DELETEITEM
wParam = (WPARAM) (int) iIndex;
lParam = 0;

```

参数

iIndex：将要删除项目的基于0的索引值。

返回值

返回一个INT值，这个值表示控件中的剩余项目数目。如果iIndex值无效，那么这个消息将返回CB_ERR。

说明

这个消息将映射到组合框控件中的CB_DELETESTRING消息。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEM_GETCOMBOCONTROL

获取子组合框控件的句柄。

```
CBEM_GETCOMBOCONTROL  
wParam = 0;  
lParam = 0;
```

返回值

返回扩展组合框控件中所包含的组合框控件的句柄。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEM_GETEDITCONTROL

获取扩展组合框控件中所包含的编辑控件部分的句柄。当某个扩展组合框控件被设置为CBS_DROPDOWN样式时, 它将使用一个编辑框。

```
CBEM_GETEDITCONTROL  
wParam = 0;  
lParam = 0;
```

返回值

如果扩展组合框控件使用的是CBS_DROPDOWN样式, 那么这个扩展组合框控件将返回其中的编辑控件的句柄。否则, 将返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_GETEXTENDEDSTYLE

获取扩展组合框控件中正在使用的扩展样式。

CBEM_GETEXTENDEDSTYLE

wParam = 0;

lParam = 0;

返回值

返回扩展组合框控件中正在使用的扩展样式的DWORD值。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_GETIMAGELIST

获取分配给扩展组合框控件的图像列表的句柄。

CBEM_GETIMAGELIST

wParam = 0;

lParam = 0;

返回值

如果执行成功，则返回分配给扩展组合框控件的图像列表的句柄，否则返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_GETITEM

获取给定扩展组合框项目的项目信息。

CBEM_GETITEM

wParam = 0;

lParam = (LPARAM)(PCOMBOBOXEXITEM)pCBItem;

参数

pCBItem：将要接收项目信息的COMBOBOXEXITEM数据结构的地址。

返回值

如果操作成功，则返回非零值；否则，返回0。

说明

如果这个消息被发送，那么其数据结构中的iItem数据成员与mask数据成员必须被设置为将要获取的目标项目索引以及信息类型，而数据结构中其他数据成员的设置则根据实际需要来决定。例如，如果需要获取文本信息，那么就必须为mask数据成员中的CBEIF_TEXT标记设置一个值，并且为cchTextMax数据成员设置一个值。如果将iItem数据成员的值设置为-1，那么将获取编辑控件中所显示的项目。

如果COMBOBOXEXITEM数据结构中mask数据成员的CBEIF_TEXT标记已经被设置了值，那么控件将把数据结构中pszText数据成员改变为指向新的文本，而不是用所需的文本填充缓冲器。应用程序不应该要求文本总是被放置在所指定的缓冲器中。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_GETUNICODEFORMAT

获取扩展组合框控件的UNICODE字符格式标记。

CBEM_GETUNICODEFORMAT

```
wParam = 0;  
lParam = 0;
```

返回值

返回控件的UNICODE格式标记。如果这个值为非零，那么控件就使用UNICODE字符；而如果这个值为0，那么控件就使用ANSI字符。

说明

关于这个消息的更多讨论，请参见CCM_GETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

CBEM_SETUNICODEFORMAT。

CBEM_HASEDITCHANGED

判断用户是否改变了扩展组合框编辑控件的文本。

CBEM_HASEDITCHANGED

```
wParam = 0;
lParam = 0;
```

返回值

如果控件中编辑框的文本发生了改变，则返回TRUE，否则将返回FALSE。

说明

如果扩展组合框控件被设置为CBS_DROPDOWN样式，那么它将使用一个编辑框控件。用户可以通过发送CBEM_GETEDITCONTROL消息方法来获取编辑框控件的窗口句柄。

在用户开始进行编辑动作时，将会接收到一个CBEN_BEGINEDIT通告消息。在编辑动作完成之后，或者焦点发生了改变之后，用户将接收到一个CBEN_ENDEDIT通告消息。如果CBEM_HASEDITCHANGED消息在CBEN_ENDEDIT通告消息之前进行发送，那么这个消息就仅仅对于判断文本是否发生了改变有用。如果这个消息在此后进行发送，那么它将返回FALSE。例如，假设用户已经开始在编辑框中进行文本的编辑，后来又改变了焦点，从而产生一个CBEN_ENDEDIT通告消息。然后，如果用户发送了一个CBEM_HASEDITCHANGED消息，那么即使是文本发生了改变，也将会返回FALSE。

CBS_SIMPLE样式不能与CBEM_HASEDITCHANGED协同正常工作。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_INSERTITEM

在扩展组合框控件中插入一个新的项目。

CBEM_INSERTITEM

```
wParam = 0;
lParam = (LPARAM)(const COMBOBOXEXITEM FAR *) lpcBItem;
```

参数

lpcBItem：COMBOBOXEXITEM数据结构的地址，在这个数据结构中包含有关于将要被插入的项目的信息。在发送消息时，iItem数据成员必须被设置为一个基于0的索引，在这个索引位置上将插入项目。如果需要在列表的最后插入某个项目，那么就应该将iItem数据成员设置为-1。

返回值

如果操作成功，则返回进行项目插入位置的索引号，否则返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEM_SETEXTENDEDSTYLE

设置扩展组合框控件的扩展样式。

```
CBEM_SETEXTENDEDSTYLE  
wParam = (WPARAM)(DWORD) dwExMask;  
lParam = (LPARAM)(DWORD) dwExStyle;
```

参数

dwExMask: 一个DWORD值, 用来表示dwExStyle参数中的哪个样式将会被影响。当然, 只有那些dwExMask参数中的扩展样式才会发生改变。如果这个参数为0, 那么dwExStyle中的所有样式都将受到影响。

dwExStyle: 一个DWORD值, 在这个参数中包含有将要为扩展组合框控件进行设置的扩展样式。

返回值

返回一个DWORD值, 在这个返回值中包含有先前在这个扩展组合框控件中所使用的扩展样式。

说明

参数dwExMask允许在不必首先获取现有样式的情况下, 修改一个或某个扩展样式。例如, 如果将CBES_EX_NOEDITIMAGE传送给参数dwExMask并且将0传送给dwExStyle, 那么CBES_EX_NOEDITIMAGE样式将被清除, 但所有其他的样式仍然保持不变。

如果试图为利用CBS_SIMPLE样式创建的扩展组合框控件设置扩展样式, 那么控件将不能进行正确的绘制。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEM_SETIMAGELIST

为扩展组合框控件设置图像列表。

```
CBEM_SETIMAGELIST  
wParam = 0;  
lParam = (LPARAM)(HIMAGELIST) himl;
```

参数

himl: 将要为扩展组合框控件设置的图像列表的句柄。

返回值

返回先前与这个控件相关联的图像列表的句柄，如果此前没有设置任何图像列表，那么将返回NULL。

说明

图像列表中图像的高度可能会导致扩展组合框控件的大小需求发生改变；因此，建议在发送这个消息之后改变控件的大小，以便确保控件的显示正确。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_SETITEM

设置扩展组合框控件中项目的属性。

CBEM_SETITEM

wParam = 0;

lParam = (LPARAM)(const COMBOBOXEXITEM FAR *) lpcCBItem;

参数

lpCBItem：COMBOBOXEXITEM数据结构的地址，在这个数据结构中包含有将要被设置的项目的信息。在这个消息被发送之后，数据结构中的mask数据成员必须被设置为使得项目属性为合法的值，并且iItem数据成员必须设置为将要修改项目的基于0的索引。如果将iItem数据成员设置为-1，那么将修改在编辑控件中显示的项目。

返回值

如果操作成功，则返回非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

CBEM_SETUNICODEFORMAT

为控件设置UNICODE字符格式的标记。这个消息允许用户在运行时改变由控件所使用的字符集，而不必重新创建控件。

CBEM_SETUNICODEFORMAT

wParam = (WPARAM)(BOOL) fUnicode;

lParam = 0;

参数

fUnicode: 判断被控件所使用的字符集。如果这个值为非零, 那么控件将使用UNICODE字符。如果这个值为0, 那么控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式标记。

说明

关于这个消息的更多讨论, 请参见CCM_SETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

CBEM_GETUNICODEFORMAT。

10.4.2 扩展组合框控件通告消息

CBEN_BEGINEDIT

当用户激活下拉列表或者在控件的编辑框中进行了单击时, 将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

CBEN_BEGINEDIT

lpmhdr = (LPMHDR) lParam;

参数

lpmhdr: NMHDR结构的地址, 在这个结构中包含有关于本通告消息的消息。

返回值

处理这个通告消息的应用程序必须返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEN_DELETEITEM

在某个项目被删除之后, 将发送这个通告消息。使这个通告消息是以WM_NOTIFY消息的形式信息发送的。

```
CBEN_DELETEITEM
    pCBEx = (PNMCOMBOBOXEX) lParam;
```

参数

pCBEx: NMCBOBOXEX数据结构的地址, 在这个数据结构中包含有关于本通告消息以及被删除的项目的信息。

返回值

处理这个通告消息的应用程序必须返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEN_DRAGBEGIN

当用户开始拖动显示在控件编辑部分的图像时, 将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
CBEN_DRAGBEGIN
    lpnmdb = (LPNMCBEDRAGBEGIN) lParam;
```

参数

lpnmdb: NMCBEDRAGBEGIN数据结构地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

所得到的返回值将被忽略。

说明

如果接受这个通告消息的应用程序实现了来自控件的拖放功能, 那么应用程序在对这个通告消息进行响应时将开始执行拖放操作。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEN_ENDEDIT

当用户完成了在编辑框中的编辑操作或者从控件的下拉列表中选择了一个项目时, 将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
CBEN_ENDEDIT
    pnmEditInfo = (PNMCBEENDEDIT) lParam;
```

参数

pnmEditInfo: NMCBEENDEDIT数据结构的地址, 在这个数据结构中包含有关于用户如何完成编辑操作的信息。

返回值

如果需要接受通告消息并且允许控件显示所选择的项目, 就应该返回FALSE。如果需要放弃所进行的编辑选择, 就应该返回TRUE。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEN_GETDISPINFO

在需要回收关于信号回叫项目的显示信息时, 就需要发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

CBEN_GETDISPINFO

pDispInfo = (PNMCOMBOBOXEX) lParam;

参数

pDispInfo: NMCOMBOBOXEX数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

处理这个通告消息的应用程序必须返回0。

说明

NMCOMBOBOXEX数据结构中包含有一个COMBOBOXEXITEM数据结构。其中的mask数据成员可以指定正在被控件所请求的信息。

如果需要向控件返回所需的信息, 那么就应该填充数据结构中适当的数据成员。如果消息处理程序将COMBOBOXEXITEM数据结构的mask数据成员设置为CBEIF_DI_SETITEM, 那么头标控件将存储这些信息并且不会进行再次的请求。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

CBEN_INSERTITEM

当有一个新项目插入到控件中时, 将发送这个通告消息。这个通告消息是以WM_NOTIFY

消息形式进行发送的。

```
CBEN_INSERTITEM
pNMInfo = (PNMCOMBOBOXEX) lParam;
```

参数

pNMInfo: NMCOMBOBOXEX数据结构的地址, 在这个数据结构中包含有关于本通告消息以及将要被插入的项目的信息。

返回值

处理这个通告消息的应用程序必须返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_SETCURSOR (扩展组合框)

告知扩展组合框控件的父窗口, 控件为了响应WM_SETCURSOR消息, 正在设置光标。这个通告消息是以WM_NOTIFY消息形式进行发送的。

```
NM_SETCURSOR
lpnm = (LPNM_MOUSE) lParam;
```

参数

lpnm: NM_MOUSE数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

如果允许控件设置光标, 则返回非零值; 如果不允许控件进行光标的设置, 则返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

10.4.3 扩展组合框控件数据结构

COMBOBOXEXITEM

在这个数据结构中包含有关于扩展组合框控件中某个项目的信息。

```
typedef struct {
    UINT    mask;
    int     iItem;
    LPCTSTR pszText;
    int     cchTextMax;
    int     iImage;
    int     iSelectedImage;
    int     iOverlay;
    int     iIndent;
    LPARAM  lParam;
} COMBOBOXEXITEM, *PCOMBOBOXEXITEM;
```

成员

mask: 这个数据成员是一系列位标记的集合，用来指定数据结构的属性，或者用来指定使用这个数据结构的操作的属性。这些位标记指定了那些合法的数据成员或者那些必须被填充值的数据成员。这个数据成员可以是以下值的组合：

位 标 记	描 述
CBEIF_DI_SETITEM	如果正在处理CBEN_GETDISPINFO消息，并且扩展组合框控件保存了所得到的信息从而不必再进行请求，那么将设置这个标记位
CBEIF_IMAGE	iImage数据成员有效或者必须被填充
CBEIF_INDENT	iIndent数据成员有效或者必须被填充
CBEIF_LPARAM	lParam数据成员有效或者必须被填充
CBEIF_OVERLAY	iOverlay数据成员有效或者必须被填充
CBEIF_SELECTEDIMAGE	iSelectedImage数据成员有效或者必须被填充
CBEIF_TEXT	pszText数据成员有效或者必须被填充

iItem: 项目基于0的索引号。

pszText: 字符缓冲器的地址，在这个字符缓冲区中可以包含或者可以接收项目的文本。如果文本信息正在被接收，那么这个数据成员就必须被设置为将要接收文本的字符缓冲器的地址。此外，必须在cchTextMax数据成员中指定缓冲器的大小。如果这个数据成员被设置为LPSTR_TEXTCALLBACK，那么控件将使用CBEN_GETDISPINFO通告消息来请求信息。

cchTextMax: pszText数据成员的长度，单位为字符数。如果文本信息正在被设置，那么这个数据成员将被忽略。

iImage: 图像列表中某个图像的基于0的索引。如果这个指定的图像没有被选中，那么它将会被项目所显示。如果这个数据成员被设置为I_IMAGECALLBACK，那么控件将利用CBEN_GETDISPINFO通告消息来请求所需的信息。

iSelectedImage: 图像列表中某个图像的基于0的索引。如果这个指定的图像被选中，那么它将会被项目所显示。如果这个数据成员被设置为I_IMAGECALLBACK，那么控件将利用CBEN_GETDISPINFO通告消息来请求所需的信息。

iOverlay: 图像列表中某后覆盖图像基于1的索引。如果这个数据成员被设置为I_IMAGECALLBACK，那么控件将利用CBEN_GETDISPINFO通告消息来请求所需的信息。

iIndent: 显示项目时进行缩排的空间大小。每次缩排大小为10个像素。如果这个数据成员

被设置为I_IMAGECALLBACK, 那么控件将利用CBEN_GETDISPINFO通告消息来请求所需的信息。

IParam: 与特定的项目相关的32位值。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMCBEENDEDIT

在这个数据结构中包含有扩展组合框控件中编辑操作的结果值。这个数据结构是与CBEN_ENDEDIT通告消息一同使用的。

```
typedef struct {
    NMHDR    hdr;
    BOOL      fChanged;
    int       iNewSelection;
    TCHAR     szText [CBEMAXSTRLEN ]; //CBEMAXSTRLEN is 260
    int       iWhy;
}NMCBEENDEDIT, *PNMCBEENDEDIT;
```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的信息。

fChanged: 用来表明控件的编辑框的内容是否发生改变的。如果编辑框中的内容发生了改变, 那么这个值将为非零, 否则将为0。

iNewSelection: 在完成编辑操作之后, 将要被选定的项目的基于0的索引。如果没有任何项目被选中, 那么这个值将为CB_ERR。

szText: 一个以0作为结束符的字符串, 在这个字符串中包含有来自控件编辑框中的文本信息。

iWhy: 用来说明产生CBEN_ENDEDIT通告消息的动的值。这个值可以是以下之一:

CBENF_DROPDOWN	表示用户激活了下拉列表。
CBENF_ESCAPE	表示用户按下了ESC键。
CBENF_KILLFOCUS	表示编辑框失去了键盘焦点。
CBENF_RETURN	表示用户按下了ENTER键, 从而完成了编辑操作。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMCBEDRAGBEGIN

在这个数据结构中包含有与CBEN_DRAGBEGIN通告消息一同使用的信息。

```
typedef struct {
    NMHDR hdr;
    int iItemid;
    TCHAR szText [CBEMAXSTLEN ];
}NMCBEDRAGBEGIN,*LPNMCBEDRAGBEGIN;
```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的信息。

iItemid: 正在被拖放项目的基于0的索引。这个值将总是为-1, 表明正在被拖动的项目是在控件的编辑部分所显示的项目。

szText: 字符缓冲器, 在这个字符串中包含有正在被拖动的项目的文本信息。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMCOMBOBOXEX

在这个数据结构中包含扩展组合框项目与通告消息一同使用时特有的信息。

```
typedef struct {
    NMHDR hdr;
    COMBOBOXEXITEM ceItem;
}NMCOMBOBOXEX,*PNMCOMBOBOXEX;
```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的信息。

ceItem: COMBOBOXEXITEM数据结构, 在这个数据结构中包含有当前通告消息所特有的项目信息。在接收到一个通告消息时, COMBOBOXEXITEM数据结构将包含控件所有者需要进行响应的信息。这个数据结构中的数据成员常常被用来作为控件所有者为了对通告消息进行响应而使用的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

第11章 创建向导

向导是一类属性页，能为用户提供一种简单而功能强大的方法，引导用户完成某个复杂的操作过程。

关于属性页面的一般讨论，请参见第21章。关于与属性页API元素的链接，请参见21.4节。

11.1 简介

向导是简化用户操作复杂性的一种关键方法。能帮助用户执行一个复杂的操作过程（例如配置应用程序），并且将这个复杂的操作过程分解为一系列简单的步骤。在这样的每个简单步骤中，用户可以提供关于自己所需要的某些描述信息，向导能够为用户提供进行选择以及输入文本的控件。

在实现方面，向导实际上是由一系列属性页来完成的。属性页本质上就是关于一个页面集合的容器，在这个容器中每个页面都是一个单独的对话框。在实现向导的过程中，用户所需要做的大部分工作与大家所熟悉的对话框编程技术相同。

一个标准的属性页能够显示许多页面，就如这些页面覆盖在其他页面上一样。每个页面在其顶部都有一个选项卡，用户可以单击页面的选项卡来选中某个页面。然后，就可以像对平常的对话框进行操作一样来与这些页面进行交互操作。图11-1显示了Microsoft Windows 2000控制面板中的日期/时间属性页。

在某一时刻，向导将显示一个页面。与选项卡方式不同的是，在向导的底部有“下一步”（Next）与“上一步”（Back）按钮。用户可以单击这些按钮来实现对这些页面进行向前与向后的导航。页面显示的顺序是由应用程序控制的，并且可以根据用户的输入信息发生顺序上的改变。图11-2显示了Windows 2000的控制面板中硬件向导上的欢迎页面。

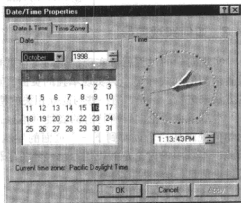


图11-1 日期/时间属性页

本章列出了创建一个Wizard97样式向导的基本过程。下一节将讨论向导实现中的一般过程。本章中的最后一节将详细描述Wiz97的实现，事实上它是一个简单的向导应用程序。

注意 在这里所进行的讨论大多数是假设为带有通用控件5.80版本及其更新版本的系统实现Wizard97样式的向导而进行的。用户可以在Windows 2000或更新版本的操作系统中使用这个样式，也可以在任何带有Internet Explorer 5或更新版本的Windows操作系统中使用这个样式。但是，在包含通用控件头文件（commctrl.h）之前，必须用#define来设置

_WIN32_IE位0x0500或更大的值。如果试图对通用控件的早期版本使用Wizard97样式，那么应用程序可能能够进行编译但不能进行正确地显示。关于如何在老版本的Windows操作系统上创建与Wizard97样式相兼容的向导的更多讨论，请参见11.3节“可向后兼容的向导”。



图11-2 Windows 2000硬件向导欢迎页面

11.2 如何实现向导

实现向导的过程就像实现常规的属性页相似。从最简单的情况来说，实现向导实际上就是设置适当的样式标记。为实现单独的页面，可以采用与在属性页中实现对话框的编程技术相同的方法来进行。但是，有一些与向导相关的特定属性页，它们几乎被所有的向导所使用。

本章重点讨论向导所特有的属性页信息，详细说明如何创建向导的属性页以及页面，并讨论如何处理从一个页面到下一个页面的导航问题。此外，本章还将讨论一些相关的设计问题。因此，本章中的讨论是假设用户已经熟悉了标准的对话框编程技术而进行的。关于属性页的一般讨论信息，请参见第21章。

一旦定义了任务并且为每个页面设计了一个用户界面，那么实现向导的基本过程就比较简单：

- 1) 为每个页面创建一个对话框模板。
- 2) 为每个页面创建一个PROPSHEETPAGE数据结构，从而定义这些页面。在这个数据结构中除了定义页面之外，还包含有指向对话框模板以及任何位图资源或其他资源的指针。
- 3) 将在上一步中创建的PROPSHEETPAGE数据结构传递给CreatePropertySheetPage，用来创建页面的HPROPSHEETPAGE句柄。
- 4) 为向导创建一个PROPSHEETPAGE数据结构，从而定义这个向导。
- 5) 将PROPSHEETPAGE数据结构传递给PropertySheet，从而显示这个向导。

6) 为每个页面实现对话框过程, 从而处理来自页面的控件以及向导按钮的通告消息, 并且处理其他Windows消息。

11.2.1 创建对话框模板

本章将讨论如何创建Wizard97样式的向导, 这个向导样式被大多数Microsoft应用程序所采用。为了保证与Wizard97样式的兼容性, 必须使得所使用的模板与Wizard97 Specification使用说明书中的模板相一致。在这个使用说明书中所定义的信息对于对话框的大小、位图大小于颜色以及控件的放置具有指导意义。

有两种基本类型的向导页面: 外部页面与内部页面。外部页面就是欢迎页面与完成页面。向导中的其他页面就是内部页面。

1. 外部页面对话框模板

所有的向导都有两个外部页面: 即欢迎页面与完成页面。这两个页面分别在向导过程的开始与结束时出现, 并且它们的基本布局方式是相同的。例如, 图11-3中显示Wizard97欢迎页面示例。

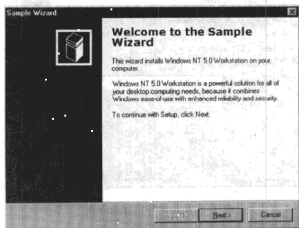


图11-3 欢迎页面示例

对于外部页面, 对话框是317×193对话框单位(dius)。外部页面将填充除了向导标题以及向导底部包含有Back(上一步)、Next(下一步)以及Cancel(取消)按钮的区域之外的所有区域。向导模板中的左面部分是为放置水印位图而预留的, 在这里不应该包含任何控件。水印位图在向导的PROPSHEETHEADER数据结构中被指定, 并且将被自动地添加到页面中。

在创建水印位图时, 应该记住的是对话框可能会增加大小(例如, 如果用户选择了更大的系统字体)。当页面变大时, 为水印位图所预留的区域也将相应地变大, 这时, 水印位图并不会为了适应这样更大的区域而将自己进行拉伸。这样, 水印位图将会以自己的原始大小位于水印位图预留区域部分左上方。那些没有被水印位图所覆盖的预留区域的剩余部分将自动地利用位图的最左上方像素的颜色进行填充。

在PROPSHEETPAGE数据结构的dwFlags数据成员中,有一些可以进行设置的、与向导相关的标志位,如下所示:

标 志 位	描 述
PSP_HIDEHEADER	必须为外部页面设置这个标志位,但不能为内部页面设置这个标志位
PSP_USEHEADERTITLE	为内部页面设置这个标志位,以便在头区域中设置一个标题
PSP_USERHEADERSUBTITLE	为内部页面设置这个标志位,以便在头区域中设置一个子标题

11.2.3 定义向导属性页

与普通的属性页相似,应该填充PROPSHEETHEADER数据结构中的数据成员,从而定义向导的属性页。这个数据结构允许指定属性页的页面从而构成向导,并且允许指定这些页面被显示的缺省顺序,同时还可以指定某些相关的参数。然后,就可以调用PropertySheet函数来运行这个向导。

在这个数据结构的dwFlags数据成员中有一系列与向导相关的标志位,如下所示:

标 志 位	描 述
PSH_USEHBMHEADER	设置这个标志位,而不是设置PSH_USEHEADER,就可以使用一个HBITMAP句柄来定义将会被放置在内部页面头区域右方的位图。应该将这个句柄分配给hbmHeader数据成员
PSH_USEHBMWATERMARK	设置这个标志位,而不是设置PSH_WATERMARK,就可以使用一个HBITMAP句柄来定义水印位图。应该将这个句柄分配给hbmWatermark数据成员
PSH_USEHEADER	设置这个标志位,就可以使用资源ID来定义头位图。可以利用MAKEINTRESOURCE来将资源ID转换为一个字符串,并且将这个字符串分配给pszbmHeader数据成员
PSH_USEPLWATERMARK	设置这个标志位,就可以为水印位图定义自己的调色板。应该将这个调色板的HPALETTE句柄分配给hplWatermark数据成员
PSH_WATERMARK	设置这个标志位,就可以使用资源ID来定义水印位图。可以利用MAKEINTRESOURCE将资源ID转换为一个字符串,并将这个字符串分配给pszbmWatermark数据成员
PSH_WIZARD97	设置这个标志位,就可以指定一个Wizard97向导样式
PSH_WIZARDCONTEXTHELP	设置这个标志位,就可以在向导的标题条上显示上下文相关的帮助按钮
PSH_WIZARDHASFINISH	设置这个标志位,就可以在所有的页面上显示“完成”(Finish)按钮。典型情况下,向导将只显示“上一步”、“下一步”或“完成”以及“取消”按钮,并且在完成页面上用“完成”按钮来取代“下一步”按钮。如果这个标志位被设置,那么将在每个页面上都显示这四个按钮

PROPSHEETHEADER数据结构的PszCaption数据成员将被忽略。相反,向导将显示在当前页面的对话框模板中所指定的标题信息。

如果用户已经为页面创建了一个关于HPROPSHEETPAGE句柄的数组,那么就应该将这个

数组分配给phpage数据成员。相反，如果创建了一个关于PROPSHEETPAGE数据结构的数组，那么就应该将这个数组分配给ppsp数据成员，并且设置dwFlags数据成员中的PSH_PROPSHEETPAGE标志位。

在下面的例程中，为了执行向导，将对一个PROPSHEETHEADER数据结构psh进行值的设置，并且调用PropertySheet函数。在Wizard97样式的向导中有水印位图与头位图，它们都由相应的资源ID所指定。在数组ahpsp中包含有所有的HPROPSHEETPAGE句柄，并且在其中定义了进行显示的缺省顺序。

```
psh.dwSize = sizeof(psh);
psh.hInstance = hInstance;
psh.hwndParent = NULL;
psh.phpage = ahpsp;
psh.dwFlags = PSH_WIZARD97 | PSH_WATERMARK | PSH_HEADER;
psh.pszbmWatermark = MAKEINTRESOURCE(IDB_WATERMARK);
psh.pszbmHeader = MAKEINTRESOURCE(IDB_BANNER);
psh.nStartPage = 0;
psh.nPages = 4;
```

```
PropertySheet(&psh);
```

11.2.4 对话框的处理过程

一旦向导开始运行，那么每个页面将需要一个对话框处理过程来处理Windows消息，特别是处理来自控件与向导的通告消息。对话框处理过程将接收到常规的WM_XXX Windows消息。几乎所有的向导都将会进行处理的三个消息是WM_INITDIALOG、WM_DESTROY与WM_NOTIFY。

1. 处理WM_INITDIALOG与WM_DESTROY

当某个页面将要被首次显示时，那么这个页面对话框处理过程将接收到一个WM_INITDIALOG消息，对这个消息的处理将使得向导能够执行任何必要的初始化任务。例如，后面将要讨论的代码实例将能够处理这个消息，从而对欢迎页面与完成页面标题中的字体进行设置。

WM_INITDIALOG消息的lParam值指向页面中PROPSHEETPAGE数据结构的一个拷贝，这时系统将忽略这个数据结构中的lParam数据成员。典型情况下，这个数据成员指向一个由应用程序所定义的共享数据结构，这个数据结构用来存储永久性数据并且将数据从一个页面传送到另一个页面。

为了保存后面将会使用的共享数据结构，应该带有GWL_USERDATA索引地调用GetWindowLong函数，从而将这个共享数据结构的指针加载到页面的用户数据中。必要时，可以使用SetWindowLong来获取共享数据结构的指针。在Wiz97示例程序中给出了一个关于如何使用共享数据结构的例子。

注意 除了能够修改PROPSHEETPAGE数据结构中的lParam数据成员之外，不能试图对其他的数据成员进行修改，否则将可能会产生不可预料的后果。

当属性页被析构时，用户将接收到一个WM_DESTROY消息。这时，系统将自动地对向导进行析构，但对这个消息的处理能够允许用户进行任何必要的清除工作。

2. 处理WM_NOTIFY

来自向导的通告消息是以WM_NOTIFY消息形式进行发送的，在相应的页面被显示之前以及在向导的任何按钮被单击时，用户将能够接收到这个消息。这个消息的lParam参数是一个指向NMHDR头数据结构的指针。通告消息的ID包含在这个数据结构的code数据成员中。大多数向导需要处理的消息是：

代 码	描 述
PSN_SETACTIVE	在页面被显示之前发送
PSN_WIZBACK	在“上一步”按钮被单击时发送
PSN_WIZNEXT	在“下一步”按钮被单击时发送
PSN_WIZFINISH	在“完成”按钮被单击时发送

(1) 处理PSN_SETACTIVE

每当某个页面将要被显示出来之前，将发送PSN_SETACTIVE通告消息。在页面被第一次访问时，PSN_SETACTIVE通告消息将跟随在WM_INITDIALOG消息之后进行发送。如果这个页面后来又被重新访问，那么它将只能接收到一个PSN_SETACTIVE通告消息。处理这个通告消息通常是为了对页面进行数据的初始化并且打开适当的按钮功能。

缺省情况下，向导将显示“上一步”、“下一步”以及“取消”按钮，并且这三个按钮都处于打开状态。如果需要关闭某个按钮或者显示“完成”按钮来取代“下一步”按钮，那么就必须发送一个PSM_SETWIZBUTTONS消息。一旦这个消息被发送，那么这个按钮状态将一直被保存，直到按钮被另一个PSM_SETWIZBUTTONS消息所修改（即使是新的页面被选中时也是如此）。典型情况下，所有的PSN_SETACTIVE处理程序都将发送这个消息，从而确保每个页面都具有正确的按钮状态。

用户可以在任何时候利用这个消息来改变按钮的状态。例如，可能会希望“下一步”按钮在初始情况下被关闭。一旦用户输入了所有必要的信息，就可以发送另一个PSM_SETWIZBUTTONS消息来打开“下一步”按钮并且让用户继续进行下一个页面的处理。

以下代码段将使用PropSheet_SetWizButtons宏来在某个内部页面被显示之前打开“上一步”按钮与“下一步”按钮：

```
case WM_NOTIFY :
{
    LPNMHDR pnmh = (LPNMHDR)lParam;
    switch(pnmh->code)
    {
        ...
        case PSN_SETACTIVE :
            ...
            //this is an interior page
            PropSheet_SetWizButtons(hwnd, PSWIZB_NEXT | PSWIZB_BACK);
            ...
    }
}
```

```

    }
    ...
}

```

(2) 处理PSN_WIZNEXT、PSNWIZBACK与PSN_WIZFINISH

当某个“下一步”按钮或者“上一步”按钮被单击时，用户将能够接收到一个PSN_WIZNEXT通告消息或PSN_WIZBACK通告消息。缺省情况下，向导将自动地按照属性页创建时所定义的顺序进入到上一个页面或者下一个页面。处理这些通告消息的常见原因就是阻止用户进行页面的切换或者覆盖缺省的页面顺序。

为了阻止用户进行页面的切换，就应该处理按钮通告消息，并且在调用SetWindowLong函数时将DWL_MSGRESULT的值设置为-1，并且返回TRUE。例如：

```

case PSN_WIZNEXT :
    ...
    //Don't go to next page yet.
    SetWindowLong(hwnd,DWL_MSGRESULT,-1);
    return TRUE;
    ...

```

如果需要覆盖标准的页面处理顺序并且转移到某个特定的页面，就应该调用SetWindowLong函数，并且在调用接口函数时将DWL_MSGRESULT的值设置为页面的对话框资源ID，然后返回TRUE。例如：

```

case PSN_WIZNEXT :
    ...
    //Go straight to the completion page.
    SetWindowLong(hwnd,DWL_MSGRESULT,IDD_FINISH);
    return TRUE;
    ...

```

当“完成”按钮或者“取消”按钮被单击时，将接收到一个PSN_WIZFINISH通告消息或者PSN_RESET通告消息。如果这两个按钮中有一个被单击，那么向导将自动地被系统所析构。但是，如果用户需要在向导被析构之前执行某些清除任务，那么就应该对这些通告消息进行处理。为了阻止向导在用户接收PSN_WIZFINISH通告消息时被析构，应该调用SetWindowLong函数，并且将DWL_MSGRESULT的值设置为TRUE，最后返回TRUE值。例如：

```

case PSN_WIZFINISH :
    ...
    //Not finished yet.
    SetWindowLong(hwnd,DWL_MSGRESULT,TRUE);
    return TRUE;
    ...

```

11.3 可向后兼容的向导

在上面的讨论中，笔者是假设用户在通用控件的5.0版本及其更新版本系统环境下实现向导的，Windows 2000及其更新版本的操作系统以及任何带有Internet Explorer 5及其更新版本的

Windows操作系统中找到通用控件的这种版本。

如果需要为通用控件的早期版本系统环境编写向导，那么前面所讨论的许多特性都不可使用，这是因为Wizard97样式中所使用的PROPSHEETHEADER数据结构与PROPSHEETHEADER数据结构中的许多数据成员都仅仅在通用控件的5.0版本及其更新版本中才提供支持。但是，仍然可以实现一个向后兼容的向导，使得这个向导在外观上与Wizard97样式的向导十分相似。为了完成这个操作，必须完成以下步骤：

- 为欢迎页面与完成页面的对话框模板添加水印位图图像。
- 使所有的模板具有相同的大小，这时没有内部页面单独系统定义的头区域。
- 在自己的模板上显式地创建内部页面的头区域。
- 尽量不要使用头位图，因为在向导的大小发生改变时，头位图将会与标题或标题发生冲突。

11.4 向导应用程序实例

本节将讨论Wiz97，它是一个简单的独立向导应用程序，用来演示关于如何实现Wizard97样式向导的过程。问题的关键在于如何创建外部页面、内部页面以及如何在它们之间进行导航。关于如何设计与实现单独的对话框，本节进行了简要的讨论。虽然在这个样本对话框中也有控件，但只有少数几个控件通告消息被处理，其目的是为了演示如何使用向导的某些特性。

在Wiz97向导中，存在一个欢迎页面、一个完成页面以及两个内部页面。在欢迎页面中有一个简单的水印位图以及两个静态控件，这两个静态控件用来存放标题文本与说明性文本（见图11-5）。

在第一个内部页面中，有由三个单选按钮和一个复选框所组成的组框（见图11-6）。需要说明的是，在初始情况下“下一步”按钮是被关闭的。只有当某个单选按钮或复选框被选中时，“下一步”按钮才被打开。如果复选框被选中，这时对“下一步”按钮的单击将直接跳转到完成页面，而不会进入第二个内部页面。

在第二个内部页面中如图11-7所示，有三个编辑框，这三个编辑框主要是为了演示的目的而提供的，它们都没有相应的处理程序。

如图11-8所示，完成页面与欢迎页面非常相似，只是在静态控件中的文本内容稍有不同，并且“下一步”按钮被“完成”按钮所取代。如果在第一个内部页面中的复选框被选中，那么单击“上一步”将会返回到这个页面。如果这个复选框没有被选中，那么单击“下一步”将进入第二个内部页面。

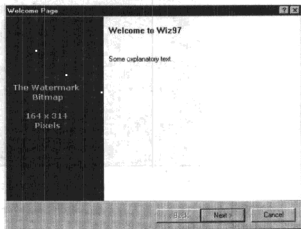


图11-5 Wiz97的欢迎页面

本节后面将讨论Wiz97应用程序的实现。为了使得代码具有更好的可读性，除了对话框处理过程之外的其他所有源代码都放置在WinMain函数中。基于相同的原因，错误检查代码被忽略。但是，用户在编写自己的向导应用程序时，应该在适当的地方使用错误检查代码。

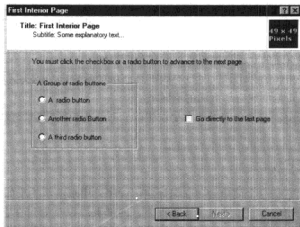


图11-6 Wiz97的第一个内部页面

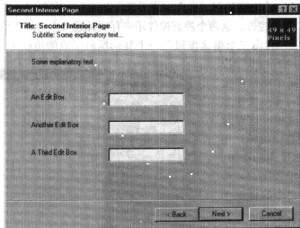


图11-7 Wiz97的第二个内部页面

需要说明的是，这个示例程序必须在带有通用控件5.80版本的系统中才能正常运行，如果试图在通用控件的更早版本中运行这个程序，将有可能导致不可预测的后果。

11.4.1 设计模板

本书不讨论关于如何创建对话框模板的机制。下面给出在设计模板时应该记住的一些关键因素：

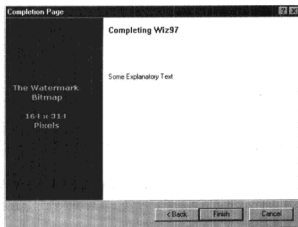


图11-8 Wiz97的完成页面

- 模板中并不包含向导中存放按钮的那部分区域。
- 对于内部页面，模板中并不包含头区域。因此，内部页面模板就小于欢迎页面与完成页面。
- 模板中并不包括水印位图，它只是为水印位图预留一片区域。水印位图是由用户在填充 PROPSHEETHEADER 数据结构时指定的。
- 最后需要记住的是，如果对话框被拉大，那么水印位图左上角位的那个像素颜色将用来进行颜色填充。

11.4.2 创建向导页面

创建向导页面的过程中主要包括：为 PROPSHEETPAGE 数据结构中的成员分配值，以及调用 CreatePropertySheetPage 来创建这个页面的 HPROPSHEETPAGE 句柄。以下代码将为名为 psp 的 PROPSHEETPAGE 数据结构分配值，从而定义欢迎页面：

```

    psp.dwSize      = sizeof( psp );
    psp.dwFlags     = PSP_DEFAULT | PSP_HIDEHEADER;
    psp.hInstance   = hInstance;
    psp.lParam      = (LPARAM) &wizdata; // shared data
                                // structure
    psp.pfnDlgProc  = IntroDlgProc;
    psp.pszTemplate = MAKEINTRESOURCE( IDD_INTRO );

    ahpsp[ 0 ] = CreatePropertySheetPage( &psp );
  
```

如果设置了 PSP_HIDEHEADER 标志位，那么将定义这个页面为外部页面，这个数据结构中的 lParam 数据成员将指向一个由应用程序所定义的 SHAREDWIZDATA 数据结构。由于 Wiz97 向导应用程序将一个指向这个数据结构的指针分配给用来创建所有页面中 PROPSHEETPAGE 数据结构 lParam 数据成员，因此 SHAREDWIZDATA 数据结构就可以用来存放共享数据。

SHAREDWIZDATA数据结构可以被用来存放由欢迎页面与完成页面所使用的HFONT, 同时还可以存放第一个内部页面中单选按钮与复选框的状态。这个数据结构定义如下:

```
typedef struct SHAREDWIZDATA {
    HFONT hTitleFont;
    BOOL IsBoxChecked;
    BOOL IsButtonClicked;
} SHAREDWIZDATA, *LP SHAREDWIZDATA;
```

欢迎页面的对话框处理过程被分配给IntroDlgProc。在设计的过程中, 最好能够保证每个页面都有自己独立的对话框处理过程。即使是一个只有静态控件的页面, 也应该具有处理向导按钮通告消息的能力。

当CreatePropertySheetPage被调用时, HPROPSHEETPAGE句柄将被分配给ahpsp[0]。ahpsp数组是一个存放HPROPSHEETPAGE句柄的数组, 可以在以后用来创建属性页。这个数组还能够决定页面将被显示的缺省顺序。

以下代码实例显示了如何创建第一个内部页面。

```
psp.dwFlags = PSP_DEFAULT | PSP_
USEHEADERTITLE | PSP_USEHEADERSUBTITLE;
psp.pszHeaderTitle = MAKEINTRESOURCE(IDS_TITLE1);
psp.pszHeaderSubTitle = MAKEINTRESOURCE(IDS_SUBTITLE1);
psp.pszTemplate = MAKEINTRESOURCE(IDD_INTERIOR1);
psp.pfnDlgProc = IntPage1DlgProc;

ahpsp [1] = CreatePropertySheetPage(&psp);
```

对于所有的页面, dwSize、hInstance以及iParam数据成员都是相同的, 并且保持不变。由于没有设置PSP_HIDEHEADER标志位, 因此这个页面是内部页面。如果对PSP_USEHEADERTITLE与PSP_USEHEADERSUBTITLE标志位进行了设置, 那么系统将能够在头区域中放置一个标题与一个子标题。这些标题是以文本字符串的形式指定的, 这些文本字符串位于本实例的字符串表中, 它们将分别被分配给pszHeaderTitle与pszHeaderSubTitle。如果将HPROPSHEETPAGE句柄分配给ahpsp[1], 那么将会使得这个页面位于缺省显示顺序的第二个位置。

用来创建第二个内部页面与完成页面的代码分别与第一个页面以及欢迎页面的代码非常相似。其不同之处在于标题、对话框处理过程等等, 它们被分配给PROPSHEETPAGE数据结构中的数据成员, 同时页面的HPROPSHEETPAGE句柄被分配给ahpsp数组中的不同元素。

11.4.3 创建属性页

创建属性页的过程与创建单独页面的过程相似。这时, 应该被PROPSHEETPAGE数据结构中的数据成员进行赋值, 然后将这个数据结构传递给PropertySheet并启动向导。以下代码将对名为psh的一个PROPSHEETHEADER数据结构进行赋值, 从而定义Wiz97的属性页。

```
psh.dwSize = sizeof(psh);
psh.hInstance = hInstance;
psh.hwndParent = NULL;
```

```

psh.phpage =      ahpsp;
psh.dwFlags =      PSH_WIZARD97 | PSH_WATERMARK | PSH_HEADER;
psh.pszbmWatermark = MAKEINTRESOURCE(IDB_WATERMARK);
psh.pszbmHeader =  MAKEINTRESOURCE(IDB_BANNER);
psh.nStartPage =   0;
psh.nPages =       4;

```

PSH_WIZARD97标志位将这个属性页设置为Wizard97向导样式。如果对PSH_WATERMARK与PSH_HEADER进行了设置,那么系统将能够向外部页面中添加一个水印位图,同时向内部页面中的头区域添加一个较小的位图。水印位图与头位图分别由它们的资源ID所标记,并且分别被分配给pszbmWatermark数据成员与pszbmHeader数据成员。只要将ahpsp数组分配给phpage数据成员,那么就能够将这些页面的句柄传递给属性页。

11.4.4 创建标题字体

在Wizard97 Specification使用说明中,欢迎页面与完成页面的标题字体是12点的Verdana Bold字体。Wiz97应用程序将创建字体,然后将HFONT句柄分类SHAREDWIZDATA共享数据结构的hTitleFont数据成员。

在进行页面的初始化时,欢迎页面与完成页面的对话框处理过程将为这个数据结构建立一个指针,然后就利用hTitleFont数据成员来设置包含标题的静态控件字体。当向导关闭时,字体也将被析构,因此字体将被创建与析构一次。以下代码演示了如何创建字体。

```

NONCLIENTMETRICS ncm = {0};
ncm.cbSize = sizeof(ncm);
SystemParametersInfo(SPI_GETNONCLIENTMETRICS, 0, &ncm, 0);
LOGFONT TitleLogFont = ncm.lfMessageFont;
TitleLogFont.lfWeight = FW_BOLD;
lstrcpyp(TitleLogFont.lfFaceName, TEXT("Verdana Bold"));

HDC hdc = GetDC(NULL);
INT FontSize = 12;
TitleLogFont.lfHeight = 0 - GetDeviceCaps(hdc, LOGPIXELSY) * FontSize /
72;
wizdata.hTitleFont = CreateFontIndirect(&TitleLogFont);
ReleaseDC(NULL, hdc);

```

11.4.5 对话框处理过程

向导被启动运行之后,应用程序的大多数其他任务是由对话框处理过程来完成的。即使是不活跃的控件,页也将需要处理一些消息(例如WM_INITDIALOG消息;来自“上一步”、“下一步”、“取消”以及“完成”按钮的通告消息)。为了简化处理任务,每个Wiz97页面都有自己独立的对话框处理过程。

1. 欢迎页面

欢迎页面中只有静态控件,因此这个页面的对话框处理过程十分简单。以下代码演示了对话框处理过程:

```

BOOL CALLBACK IntroDlgProc (
    HWND hwndDlg,
    UINT uMsg,
    WPARAM wParam,
    LPARAM lParam
)
{
    LPSHAREDWIZDATA pdata = (LPSHAREDWIZDATA)GetWindowLong(hwndDlg,
    GWL_USERDATA);

    switch (uMsg)
    {
        case WM_INITDIALOG :
        {
            pdata = (LPSHAREDWIZDATA)((LPPROPSHEETPAGE)lParam)->lParam;

            SetWindowLong(hwndDlg, GWL_USERDATA, (DWORD_PTR)pdata);

            HWND hwndControl = GetDlgItem(hwndDlg, IDC_TITLE);
            SetWindowFont(hwndControl, pdata->hTitleFont, TRUE);
            break;
        }

        case WM_NOTIFY :
        {
            LPNMHDR lpnm = (LPNMHDR)lParam;

            switch (lpnm->code)
            {
                case PSN_SETACTIVE ://Enable the Next button
                    PropSheet_SetWizButtons(GetParent(hwndDlg), PSWIZB_NEXT);
                    break;
                default :
                    break;
            }
        }
        break;

    default:
        break;
    }
    return 0;
}

```

当一个页面的对话框处理过程被首次调用时，它将接收到一条WM_INITDIALOG消息。这个消息的处理程序将从页面PROPSHEETPAGE数据结构的lParam数据成员中提取出指向SHAREDWIZDATA共享数据结构的指针。然后，它将利用SetWindowLong将这个指针分配给页

面的用户数据。在对这个对话框处理过程的每次后续调用中，将使用GetWindowLong来提取SHAREDWIZDATA指针以供后用。在存储这个指针之后，消息处理程序将使用SHAREDWIZDATA数据结构的hTitleFont数据成员来为包含本标题的静态控件设置字体。

在接收到WM_INITDIALOG消息之后，后来每次这个页面被选中，它的对话框处理过程将接收到一条PSN_SETACTIVE通告消息。这个通告消息是在相应的页面被显示之前发送的，用来进行初始化动作。欢迎页面的PSN_SETACTIVE通告消息处理程序使用PropSheet_SetWizButtons宏来发送PSM_SETWIZBUTTONS消息，从而打开“下一步”按钮。由于向导能够自动地显示下一个页面，因此“下一步”按钮的通告消息将不会被处理。虽然在以上代码中没有表示出来，但事实上示例代码中是有标记按钮处理程序作为占位符的。

2. 内部页面

初始时，第一个内部页面的显示中“下一步”按钮是被关闭的，只有在某个单选按钮或复选框被选中之后，这个按钮才打开。如果用户选择了复选框，那么单击“下一步”按钮将直接进入完成页面。以下代码演示了内部页面的对话框处理过程：

```

BOOL CALLBACK IntPageDlgProc (
    HWND hwndDlg,
    UINT uMsg,
    WPARAM wParam,
    LPARAM lParam
)
{
    LPSHAREDWIZDATA pdata = (LPSHAREDWIZDATA)GetWindowLong
(hwndDlg, GWL_USERDATA);

    switch (uMsg)
    {
        case WM_INITDIALOG :
        {
            pdata = (LPSHAREDWIZDATA)((LPPROPSHEETPAGE)lParam)->lParam;
            SetWindowLong(hwndDlg, GWL_USERDATA, (DWORD_PTR)pdata);
            break;
        }

        case WM_COMMAND :
        {
            switch (LOWORD (wParam))
            {
                case IDC_CHECK1 :
                    pdata->IsBoxChecked = !(pdata->IsBoxChecked);
                    break;

                case IDC_RADIO1 :

                case IDC_RADIO2 :

                case IDC_RADIO3 :
            }
        }
    }
}

```

```
        pdata->IsButtonClicked = TRUE;
        break;
    default :
        break;
    }

    if(pdata->IsBoxChecked || pdata->IsButtonClicked)
    {
        PropSheet_SetWizButtons(GetParent(hwndDlg),
        PSWIZB_BACK | PSWIZB_NEXT);
    }

    else
    {
        PropSheet_SetWizButtons(GetParent(hwndDlg), PSWIZB_BACK);
    }
    break;

case WM_NOTIFY :
{
    LPNMHDR lpnm = (LPNMHDR)lParam;

    switch (lpnm->code)
    {
        case PSN_SETACTIVE ://Enable the appropriate buttons.
            if(pdata->IsBoxChecked || pdata->IsButtonClicked)
            {
                PropSheet_SetWizButtons(GetParent
                (hwndDlg), PSWIZB_BACK | PSWIZB_NEXT);
            }

            else //Otherwise, only enable Back.
            {
                PropSheet_SetWizButtons(GetParent(hwndDlg), PSWIZB_BACK);
            }
            break;

        case PSN_WIZNEXT :
            if(pdata->IsBoxChecked)
            {
                SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_END);
                return TRUE;
            }
            break;

        default :
            break;
    }
}
```



```

    }
    break;
default:
    break;
}
return 0;
}

```

WM_INITDIALOG处理程序将提取指向SHAREDWIZDATA共享数据结构的指针，并且将这个指针分配给页面的用户数据。初始情况下，PSN_SETACTIVE处理程序仅仅打开“上一步”按钮，而只有在用户选择了一个单选按钮或复选框之后，“下一步”按钮才被打开。如果复选框被选中，那么PSN_WIZNEXT处理程序将使用SetWindowLong直接跳转到最后一个页面。

SHAREDWIZDATA共享数据结构中的IsBoxChecked数据成员可以用来存储复选框的状态。IsBoxChecked数据成员的值用来表示是否有一个单选按钮被选中。在SHAREDWIZDATA数据结构中存放这个状态参数可以有两个作用：其一就是使得状态信息能够保持不变，从而使得页面被再次访问时这个状态有效；其二就是使得状态信息对其他页面也是有效的。

第二个内部页面中并没有实现其中三个编辑框中的任何一个，因此它的对话框处理过程最小。与其他对话框处理过程一样，这个页面的WM_INITDIALOG消息处理程序将把指向SHAREDWIZDATA共享数据结构的指针分配给页面的用户数据。而它的PSN_ACTIVE通告消息处理程序将打开“下一步”按钮与“上一步”按钮。

11.5 完成页面

完成页面的对话框处理过程与欢迎页面的对话框处理过程相似。下面给出了完成页面的对话框处理代码：

```

BOOL CALLBACK EndDlgProc (
    HWND hwndDlg,
    UINT uMsg,
    WPARAM wParam,
    LPARAM lParam
)
{
    LPSHAREDWIZDATA pdata = (LPSHAREDWIZDATA)GetWindowLong
(hwndDlg, GWL_USERDATA);

    switch (uMsg)
    {
        case WM_INITDIALOG :
        {
            pdata = (LPSHAREDWIZDATA) ((LPPROPSHEETPAGE)lParam)->lParam;
            SetWindowLong(hwndDlg, GWL_USERDATA, (DWORD_PTR)pdata);

            HWND hwndControl = GetDlgItem(hwndDlg, IDC_TITLE);

```

```

        SetWindowFont(hwndControl, pdata->hTitleFont, TRUE);
        break;
    }

    case WM_NOTIFY :
    {
        LPNMHDR lpnm = (LPNMHDR)lParam;

        switch (lpnm->code)
        {
            case PSN_SETACTIVE :
                PropSheet_SetWizButtons(GetParent(hwndDlg),
                PSWIZB_BACK | PSWIZB_FINISH);
                break;

            case PSN_WIZBACK :

                if(pdata->IsBoxChecked)
                {
                    SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_INTERIOR1);
                    return TRUE;
                }
                break;

            default :
                break;
        }
    }
    break;

default:
    break;
}
return 0;
}

```

与欢迎页面一样，WM_INITDIALOG处理程序将提取SHAREDWIZDATA共享数据结构，并且将它的指针存放在页面的用户数据中。然后，使用hTitleFont数据成员来设置包含标题的静态控件的字体。PSN_SETACTIVE处理程序将打开“上一步”按钮与“完成”按钮。

PSN_WIZBAK处理程序将检查SHAREDWIZDATA共享数据结构的IsBoxChecked数据成员，查看第二个内部页面中的复选框是否被选中。若是，处理程序将返回到第一个内部页面，而不是回到第二个内部页面。只要这个复选框被选中，用户将永远不能见到第二个内部页面。

由于不需要进行任何清除处理，因此没有PSN_WIZFINISH处理程序。在“完成”按钮被单击之后，向导将自动地被析构。在向导被析构之后，控件将返回到WinMain函数，这个函数将使用DeleteObject函数来析构标题字体，最后将返回到调用位置。

第12章 日期与时间检出发控件

日期与时间检出发 (date and picker, DTP) 控件能够提供一个简单而具有引导性的界面, 利用这个界面可以实现与用户进行日期信息与时间信息的交换。例如, 利用DTP控件, 可以要求用户输入一个日期, 然后非常方便地获取他/她所输入的信息。

12.1 关于日期与时间检出发控件

日期与时间检出发 (DTP) 控件在4.70以及后续的Comctl32.dll版本中实现。如果需要创建DTP控件, 就应该调用CreateWindowEx函数并且将DATETIMEPICK_CLASS指定为窗口类。当日期与时间检出发器类从通用控件动态链接库 (DLL) 中进行加载时, 这个类将被注册。应该调用InitCommonControlsEx函数, 并且指定INITCOMMONCONTROLSEX数据结构中的ICC_DATE_CLASSES位标记来完成对这个类的注册。

注意 Windows并不支持1601年之前的日期。关于详细信息, 请参见FILETIME。DTP控件是基于Gregorian日历进行日期计算的, 这个日历从1753年被引入。它不能计算在1753年之前就被使用的Julian日历。

12.1.1 日期与时间检出发用户界面

日期与时间检出发控件的客户区中显示日期信息与时间信息, 并且作为用户进行这些信息修改的界面。在这个控件的显示中, 包含有控件的格式字符串定义域。此外, 如果使用了DTS_SHOWNONE样式, 那么控件还将显示一个复选框。

控件中的每个域都显示存储在这个控件内部的日期与时间信息部分。用户可以单击某个域, 从而设置键盘焦点, 然后通过键盘输入信息以改变相应域中的信息。DTP控件将自动地根据用户的输入信息更新内部日期信息。DTP控件能够认识以下形式的合法输入:

输入信息分类	描 述
箭头键	控件接收箭头键, 从而在控件中的域之间进行导航并且改变值。用户可以按下左箭头键或右箭头键来在控件中的域之间进行移动。如果用户希望在一个给定的方向上移过一个域, 那么键盘焦点将转移到控件所在域的另一端。向上箭头键与向下箭头键能够在当前域中对值进行增减
End键与Home键	控件接受VK_END与VK_HOME虚拟键, 从而将当前域中的值分别改变为值的上限与下限
功能键	F2键可以激活编辑模式。F4键将使得控件显示一个下拉列表样式的月历控件 (如果用户按下ALT + DOWN ARROW (向下箭头) 也能够实现此功能)
数字键	控件能够接受以两个字符的形式输入的数字。如果所输入的数字非法 (例如在月份中输入14), 那么控件将拒绝这个输入并且将这个域的值重新设置为先前值
加号键与减号键	控件能够接受来自数字键盘的VK_ADD与VK_SUBTRACT虚拟键, 从而对当前域中的值进行自加与自减

DTP控件并不使用DTS_UPDOWN样式来显示箭头按钮。如果用户单击了这个按钮,那么月历控件将被下拉出来。用户可以单击这个月历中的某个区域,从而选择一个特定的日期。

12.1.2 日期与时间检出器控件的样式与格式

在日期与时间检出器控件中有几个样式可以用来决定控件的外观与行为。在创建日期与时间检出器控件时,应该用CreateWindowEx的dwStyle参数来指定控件的样式。在创建控件之后,如果需要获取或改变窗口样式,就应该使用GetWindowLong与SetWindowLong函数来完成此动作。

1. 预先设置格式

有三个预先设置的格式可用来显示日期,以及一个预先设置的格式可用来显示时间。用户可以通过选择以下窗口样式之一来完成这些设置:

DTS_LONGDATEFORMAT

这个样式的显示形式是这样的:“Friday, April 19, 1996”。

DTS_SHORTDATEFORMAT

这个样式的显示形式是这样的:“4/19/96”。

DTS_SHORTDATECENTURYFORMAT

这个样式在5.80版本中有效。它的显示形式是这样的:“4/19/1996”。

DTS_TIMEFORMAT

这个样式的显示形式是这样的:“5:31:42 PM”。

2. 自定义格式

DTP控件必须基于格式字符串来决定其中域信息的显示方式。如果预先设置的格式不能满足自己的要求,那么用户就可以定义自己的格式字符串,从而创建一个自定义格式。对于应用程序,自定义格式能够提供更大的显示灵活性。它们允许用户指定控件显示其中域信息的顺序。在向用户请求信息时,可以包括信息文本部分以及信号回叫域。一旦格式字符串被创建,还必须利用DTM_SETFORMAT消息将这个格式字符串分配给DTP控件。

格式字符串

DTP格式的字符串中包含有一系列表示特定信息的元素和其显示格式。这些元素将按照其在格式字符串中的顺序来显示。

日期与时间格式元素会被实际的日期与时间代替。下列一组字符定义了这些格式。

元 素	描 述
"d"	由一位数字或者两位数字表示的日信息
"dd"	由两位数字表示的日信息。如果这个日信息只有一个有效数字位,那么必须在这个数字位的前面补上一个0
"ddd"	由三个字符表示的星期信息简写
"dddd"	完整的星期信息名
"h"	由一位数字或者两位数字表示的12小时格式小时信息
"hh"	由两位数字表示的12小时格式小时信息。如果这个小时信息只有一个有效数字位,那么必须在这个数字位的前面补上一个数字0

(续)

元 素	描 述
"H"	由一位数字或者两位数字表示的24小时格式小时信息
"HH"	由两位数字表示的24小时格式小时信息。如果这个小时信息只有一个有效数字位,那么必须在这个数字位的前面补上一个数字 0
"m"	由一位数字或者两位数字表示的分钟信息
"mm"	由两位数字表示的分钟信息。如果这个分钟信息只有一个有效数字位,那么必须在这个数字位的前面补上一个0
"M"	由一位数字或者两位数字表示的月份信息
"MM"	由两位数字表示的月份信息。如果这个月份信息只有一个有效数字位,那么必须在这个数字位的前面补上一个0
"MMM"	由三个字符表示的简写月份信息
"MMMM"	月份信息的全名
"t"	由一个字母表示的AM/PM简写(例如,AM被显示为"A"形式)
"tt"	由两个字母表示的AM/PM简写(例如,AM被显示为"AM"形式)
"yy"	年份信息的最后两个数字位(例如,1996将被显示为96)
"yyyy"	年份信息的全名(例如,1996被显示为"1996")

为了使得上述信息具有更好的可读性,可以用单引号标记将向格式字符串中添加的文本包括起来。空格符与标点符号字符将不包含在单引号标记中。

注意 没有被单引号标记所分割的非格式化字符将有可能会使得DTP控件产生不可预测的显示结果。

例如,如果需要按照格式“Today is:04:22:31 Tuesday Mar 23,1996”的形式来显示当前日期,那么相应的格式字符串就应该是“Today is: 'hh': 'm': 's' dddd MMM dd, 'yyyy'”。如果需要在文本体中包含一个单引号标记,就应该使用两个连续的单引号标记。例如,格式字符串“Don't forget'MMM dd,'yyyy'”将产生以下形式的显示格式: Don't forget Mar 23, 1996。对于逗号,可以不使用单引号标记,因此,格式字符串“Don't forget'MMM dd, yyyy”也是合法的,所产生的结果相同。

3. 信号回叫域

除了可以定义标准的格式字符与文本体之外,还可以将显示中的某些部分定义为信号回叫域。这些域可以用来向用户查询有关信息。为了声明一个信号回叫域,必须在格式字符串中的任何位置包含一个或多个“X”字符(它的ASCII代码为88)。如果反复地使用“X”字符,用户就可以创建拥有唯一标识的信号回叫域。这样,格式字符串“XX dddd MMM dd”中将包含两个唯一的信号回叫域:“XX”与“XXX”。与其他DTP控件域一样,信号回叫域也是基于它们在格式字符串中的位置按照从左到右的顺序显示的。

当DTP控件对格式字符串进行分解并且遇到一个信号回叫域时,它将发送DTN_FORMAT与DTN_FORMATQUERY通告消息。这时,与信号回叫域相对应的格式字符串元素将被包含在通告消息中,从而允许接收这个通告消息的应用程序来判断被查询的是哪个信号回叫域。控件的所有者必须对这些通告消息进行响应,从而确保自定义信息能够正确地显示。

12.1.3 日期与时间检出器控件通告消息

当日期与时间检出器控件接收到用户输入信息或者正在处理并响应信号回调域时，它将发送通告消息。控件的父窗口将以WM_NOTIFY消息的形式接收这些通告消息。

通告消息	描述
DTN_CLOSEUP	表明下拉列表形式的月历将被删除
DTN_DATETIMECHANGE	表明在DTP控件中发生了某个改变
DTN_DROPDOWN	表明下拉列表形式的月历将被显示
DTN_FORMAT	请求将要在信号回调域的格式字符串部分中显示的文本信息
DTN_FORMATQUERY	请求将要在信号回调域中显示的文本的最大可允许文本大小信息
DTN_USERSTRING	告知用户在控件中结束了编辑操作。这个通告消息仅仅被使用DTS_APPCANPARSE样式的DTP控件所发送
DTN_WMKEYDOWN	告知用户已经在DTP控件的信号回调域中按下了一个键

12.2 使用日期与时间检出器控件

本节将给出实现一个日期与时间检出器控件的信息及其代码。

12.2.1 创建日期与时间检出器控件

如果需要创建日期与时间检出器控件，那么就应该使用CreateWindowEx函数，并且将DATETIMEPICK_CLASS指定为窗口类。首先必须调用InitCommonControlsEx函数，并且指定INITCOMMONCONTROLSEX数据结构中的ICC_DATE_CLASSES标记位，从而完成对这个窗口类的注册。

以下代码将在一个现有的非模态对话框中创建DTP控件。这个控件使用DTS_SHOWNONE样式，从而允许用户在控件中模拟日期的不激活过程。

```
// CreateDatePick creates a DTP control within a dialog box.
// Returns the handle to new DTP control if successful,
// or NULL otherwise.
//
// hwndMain -The handle to the main window.
// g_hinst -global handle to the program instance.
HWND WINAPI CreateDatePick(hwndMain)
{
    HWND hwndDP = NULL;
    INITCOMMONCONTROLSEX icex;

    icex.dwSize = sizeof(icex);
    icex.dwICC = ICC_DATE_CLASSES;

    InitCommonControlsEx(&icex);

    hwndDlg = CreateDialog (g_hinst,
```

```

MAKEINTRESOURCE(IDD_DIALOG1),
hWndMain,
DlgProc);

if (hWndDlg)
{
    hWndDP = CreateWindowEx(0,
        DATETIMEPICK_CLASS,
        'DateTime',
        WS_BORDER | WS_CHILD | WS_VISIBLE | DTS_SHOWNONE,
        20, 50, 220, 20,
        hWndDlg,
        NULL,
        g_hInst,
        NULL);
    return (hWndDP);
}

```

12.2.2 处理日期与时间检出器通告消息

以下代码将分别调用应用程序定义的DoDateTimeChange、DoFormatQuery、DoFormat与DoWMKeydown函数来处理DTN_DATETIMECHANGE、DTN_FORMAT、DTN_FORMATQUERY与DTN_WMKEYDOWN通告消息。

本章中的其他部分将讨论关于这些通告消息的更多信息。详细信息请参见12.2.4节以及12.2.3节的内容。

```

BOOL WINAPI DoNotify(HWND hWnd, WPARAM wParam, LPARAM lParam)
{
    LPNMHDR hdr = (LPNMHDR)lParam;

    switch(hdr->code){
        case DTN_DATETIMECHANGE:{
            LPNMDATETIMECHANGE lpChange = (LPNMDATETIMECHANGE)lParam;
            DoDateTimeChange(lpChange);
        }
        break;

        case DTN_FORMATQUERY:{
            LPNMDATETIMEFORMATQUERY lpDTFQuery = (LPNMDATETIMEFORMATQUERY)lParam;

            //Process DTN_FORMATQUERY to ensure that the
            //control displays callback information
            //properly.
            DoFormatQuery(hdr->hWndFrom, lpDTFQuery);
        }
        break;

        case DTN_FORMAT:{
            LPNMDATETIMEFORMAT lpNMFormat = (LPNMDATETIMEFORMAT)lParam;

```

```

        // Process DTN_FORMAT to supply information
        // about callback fields (fields) in the DTP
        // control.
        DoFormat(hdr->hwndFrom, lpNMFormat);
    }
    break;

case DTN_WMKEYDOWN: {
    LPNMDATETIMEWMKEYDOWN lpDTKeystroke =
        (LPNMDATETIMEWMKEYDOWN)lParam;

    // Process DTN_WMKEYDOWN to respond to a user
    // keystroke in a callback field.
    DoWMKeydown(hdr->hwndFrom, lpDTKeystroke);
}
    break;
}
// All of the above notifications require the owner to
// return zero.
return FALSE;
}

```

12.2.3 处理DTN_DATETIMECHANGE通告消息

当日期与时间检出器控件发生了任何改变时，它将发送DTN_DATETIMECHANGE消息。如果用户改变了控件中的某个域或者改变了控件的复选框状态（对于DTS_SHOWNONE），那么将产生这个消息。

以下是一个应用程序所定义的函数，可以用来更新对话框中的一个静态控件。这个静态控件中的文本将会发生改变，从而反映DTP控件的当前状态：

```

void WINAPI DoDateTimeChange(LPNMDATETIMECHANGE lpChange)
{
    //If the user has unchecked the DTP's check box,
    //change the text in a static control to show the
    //appropriate message.
    //
    //g_hwndDlg -a program-global address of a dialog box.

    if(lpChange->dwFlags &= GDT_NONE)
        SetDlgItemText(g_hwndDlg, IDC_STATUS, 'Disabled');
    else
        SetDlgItemText(g_hwndDlg, IDC_STATUS, 'Active');
}

```

12.2.4 在DTP控件中支持信号回叫域

如果希望在应用程序的日期与时间检出器控件中使用信号回叫域，那么应用程序就必须能

够处理DTN_FORMATQUERY、DTN_FORMAT与DTN_WMKEYDOWN通告消息。关于信号回叫域的更多信息，请参见12.1.2节中的“信号回叫域”。

本节将讨论应用程序如何处理这些通告消息，同时给出几个由应用程序所定义函数的源代码。下表列出了这些函数将要处理的通告消息：

通告消息	处理函数
DTN_FORMAT	DoFormat
DTN_FORMATQUERY	DoFormatQuery
DTN_WMKEYDOWN	DoWMKeydown

1. DoFormatQuery应用程序定义函数

为了获取控件中某个信号回叫域中文本的最大大小信息，日期与时间检出器控件将发送一个DTN_FORMATQUERY通告消息。对这个通告消息的处理，将能够确保所有的域都被正确地显示。以下由应用程序所定义的DoFormatQuery函数将能够处理DTN_FORMATQUERY通告消息，其方法是计算给定信号回叫域中最长字符串的宽度：

```
//
// DoFormatQuery processes DTN_FORMATQUERY messages to
// ensure that the DTP control displays callback field
// properly.
//
void WINAPI DoFormatQuery(
    HWND hwndDP,
    LPNMDATETIMEFORMATQUERY lpDTPQuery)
{
    HDC hdc;
    HFONT hFont, hOrigFont;

    //Prepare the device context for GetTextExtentPoint32call.
    hdc = GetDC(hwndDP);

    hFont = FORWARD_WM_GETFONT(hwndDP, SendMessage);

    if(!hFont)
        hFont = (HFONT)GetStockObject(DEFAULT_GUI_FONT);
    hOrigFont = SelectObject(hdc, hFont);

    //Check to see if this is the callback segment
    //desired.If so,use the longest text segment to
    //determine the maximum width of the callback field,
    //and then place the information into the
    //NMDATETIMEFORMATQUERY structure.
    if(!strcmp("XX", lpDTPQuery->pszFormat))
        GetTextExtentPoint32(hdc,
            "366", //widest date string
            3,
```

```

        &lpDTFQuery->szMax);

//Reset the font in the device context;then release
//the context.
SelectObject(hdc,hOrigFont);
ReleaseDC(hwndDP,hdc);
}

```

2. DoFormat应用程序定义函数

为了获取将要在控件的信号回调域中显示的文本，日期与时间检出器控件将发送DTN_FORMAT通告消息。对这个通告消息的处理，将允许DTP控件显示那些本来不支持的信息。

以下由应用程序所定义的DoFormat函数将处理DTN_FORMAT通告消息，其方法是为信号回调域提供文本字符串。DoFormat将调用应用程序所定义的GetDayNum函数来请求将在信号回调字符串中使用的日数据。

```

//DoFormat processes DTN_FORMAT to provide the text for a
//callback field in a DTP control.In this example,the
//callback field contains a value for the day of year.
//The function calls the application-defined function
//GetDayNum (below)to retrieve the correct value.
//
void WINAPI DoFormat(HWND hwndDP,LPNMDATEFORMATlpDTFormat)
{
    wsprintf(lpDTFormat->szDisplay,"%d",GetDayNum(&lpDTFormat->st));
}

```

3. GetDayNum应用程序定义函数

为了请求基于当前日期的日数据，由应用程序所定义的函数DoFormat将调用以下由应用程序所定义的GetDayNum函数。GetDayNum函数返回一个INT值，用来表示本年的当前日，其范围是0~366。在这个函数的处理过程中，它将调用另一应用程序所定义的函数IsLeapYr。

```

//
//GetDayNum is an application-defined function that
//retrieves the correct day of the year value based on
//the contents of a given SYSTEMTIME structure.This
//function calls the IsLeapYr function to check if the
//current year is a leap year.The function returns an
//integer value that represents the day of year.
//
int WINAPI GetDayNum(SYSTEMTIME *st)
{
    int iNormYearAccum [] = {31,59,90,120,151,181,
        212,243,273,304,334,365};

    int iLeapYearAccum [] = {31,60,91,121,152,182,
        213,244,274,305,335,366};
}

```

```

int iDayNum;

if(IsLeapYr(st->wYear))
    iDayNum=iLeapYearAccum [st->wMonth-2 ]+st->wDay;
else
    iDayNum=iNormYearAccum [st->wMonth-2 ]+st->wDay;

return (iDayNum);
}

```

4. IsLeapYr应用程序定义函数

为了判断当前年是否为闰年，由应用程序所定义的函数GetDayNum将调用IsLeapYr函数。如果当前年是闰年，那么IsLeapYr将返回一个值为TRUE的BOOL值；否则，将返回FALSE。

```

//
//IsLeapYr determines if a given year value is a leap
//year.The function returns TRUE if the current year is
//a leap year,and FALSE otherwise.
//
BOOL WINAPI IsLeapYr(int iYear)
{
    int iQuotient;
    BOOL fLeapYr =FALSE;

    //If the year is evenly divisible by 4 and not by 100,
    //then this is a leap year.
    if(!(iYear%4)&&(iYear%100))
        fLeapYr =TRUE;
    else{
        //If the year is evenly divisible by 4 and 100,
        //then check to see if the quotient of the year
        //divided by 100 is also evenly divisible by 4.
        //If it is,then this is a leap year.
        if(!(iYear%4)&&!(iYear%100)){
            iQuotient =iYear/100;
            if(!(iQuotient%4))fLeapYr =TRUE;
        }
    }
    return (fLeapYr);
}

```

5. DoWMKeydown应用程序定义函数

为了报告用户已经在信号回叫域中进行了输入动作，日期与时间检出器控件将发送DTN_WMKEYDOWN消息。对这个消息的处理将使得能够模拟由标准DTP域所提供的键盘响应动作或者提供自定义的键盘响应动作。以下由应用程序所定义的函数DoWMKeydown就是一个关于如何处理DTN_WMKEYDOWN通告消息的例子：

```
//
// DoWMKeydown increments or decrements the day of month
// according to user keyboard input.
//
void WINAPI DoWMKeydown(
    HWND hwndDP,
    LPNMDATETIMEWMKEYDOWN lpDTKeystroke)
{
    int delta = 1;

    if(!strcmp(lpDTKeystroke->pszFormat, "XX")){
        switch(lpDTKeystroke->nVirtKey){
            case VK_DOWN:
            case VK_SUBTRACT:
                delta = -1; //fall through

            case VK_UP:
            case VK_ADD:
                lpDTKeystroke->st.wDay += delta;
                break;
        }
    }
}
```

12.3 日期与时间检出器控件样式

下面所列出的窗口样式是日期与时间检出器控件所特有的样式。那个用来定义的显示格式的DTS_XXXFORMAT样式不能被组合使用。如果没有所列出的样式能够满足设计需求，那么就应该使用DTM_SETFORMAT消息来定义一个自定义样式：

DTS_APPCANPARSE

允许控件所有者对用户输入进行解析并采取必要的动作。这个样式使得用户在按下F2键之后能够在控件的客户区中进行编辑动作。当用户完成编辑动作之后，控件将发送DTN_USERSTRING通告消息。

DTS_LONGDATEFORMAT

以长格式显示日期信息。这个样式的缺省格式字符串由LOCALE_SLONGDATEFORMAT所定义，它所产生的输出形式为“Friday, April 19, 1996”。

DTS_RIGHTALIGN

下拉列表形式的月历将在控件中进行右对齐，而不是进行缺省方式的左对齐。

DTS_SHORTDATEFORMAT

以短格式显示日期信息。这个样式的缺省格式字符串由LOCALE_SSHORTDATE所定义，它所产生的输出形式为“4/19/96”。

DTS_SHORTDATECENTURYFORMAT	在5.80版本中有效。这个样式与DTS_SHORTDATEFORMAT样式相似,只是在这个样式中年份信息是以四位的域进行显示的。这个样式的缺省格式字符串是基于LOCALE_SSHORTDATE定义的,它所产生的输出形式为“4/19/1996”。
DTS_TIMEFORMAT	这个样式用来显示时间。它的缺省格式字符串是由LOCALE_STIMEFORMAT定义的,所产生的输出形式为“5:31:42 PM”。
DTS_SHOWNONE	有可能会存在这种情况:在控件中当前没有选中任何日期。对于这种样式,控件将显示一个复选框,用户可以在输入或选择某个日期之后选中这个复选框。在这个复选框被选中之前,应用程序将不能从控件中获取日期信息,这是因为控件当前还没有日期信息。这个状态可以利用DTM_SETSYSTEMTIME消息来进行设置,也可以用DTM_GETSYSTEMTIME消息来查询这项设置状态。
DTM_UPDOWN	将一个增减数控件放置在DTP控件的右边,用来修改日期时间值。这个样式可以用来取代缺省的拉列表月历样式。

12.4 日期与时间检出器参考

12.4.1 日期与时间检出器控件消息

DTM_GETMCCOLOR

获取日期与时间检出器控件中月历部分的颜色。可以显式地发送这个消息,也可以使用DateTime_GetMonthCalColor宏来发送这个消息。

```
DTM_GETMCCOLOR
wParam = (WPARAM)(INT) iColor;
lParam = 0;
```

参数

iColor: 用来指定哪个月历颜色将被获取的INT值,这个值可以是以下值之一:

MCSC_BACKGROUND	获取在月份之间所显示的背景色。
MCSC_MONTHBK	获取在月份内部显示的背景色。
MCSC_TEXT	获取在月份内部文本上显示的颜色。

MCSC_TITLEBK	获取在月历的标题上显示的背景色。
MCSC_TITLETEXT	获取在月历的标题上显示的文本颜色。
MCSC_TRAILINGTEXT	获取用来显示第一日与最后一日文本的颜色。这两个日子是在当前月历中上一个月份与下一个月份之间所出现的日。

返回值

如果操作成功，则返回用来表示月历中特定部分的颜色设置，否则将返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DTM_GETMCFONT

获取日期与时间检出器控件等子月历控件当前正在使用的字体。可以显式地发送这个消息，也可以利用DateTime_GetMonthCalFont宏来发送这个消息。

```
DTM_GETMCFONT
wParam = 0;
lParam = 0;
```

返回值

返回当前字体句柄的HFONT值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DTM_GETMONTHCAL

获取日期与时间检出器中子月历控件的句柄。可以显式地发送这个消息，也可以利用DateTime_GetMonthCal宏来发送这个消息。

```
DTM_GETMONTHCAL
```

```
wParam = 0;
lParam = 0;
```

返回值

如果操作成功，则返回DTP控件的子月历控件句柄，否则返回NULL。

说明

当用户单击下拉列表箭头时，DTP控件将创建一个子月历控件。如果不再需要这个月历控件，那么它将被析构。因此应用程序不能使用静态句柄来存放DTP控件的子月历控件。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

DTN_CLOSEUP、DTN_DROPDOWN。

DTM_GETRANGE

为日期与时间检出器控件获取系统当前所允许的最大与最小时间。可以显式地发送这个消息，也可以利用DateTime_GetRange宏来发送这个消息。

```
DTM_GETRANGE
wParam = 0;
lParam = (LPARAM) lpSysTimeArray;
```

参数

lpSysTimeArray：SYSTEMTIME数据结构的拥有两个元素的数组的地址。

返回值

返回一个DWORD值，这个值是GDTR_MIN或GDTR_MAX的组合。如果GDTR_MIN的值被设置，那么SYSTEMTIME数组中的第一个元素将包含最小的允许时间。如果GDTR_MAX的值被设置，那么SYSTEMTIME数组的第二个元素的值将包含最大的允许时间。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DTM_GETSYSTEMTIME

从日期与时间检出器控件中获取当前选择的时间，并将这个时间放置到指定的SYSTEMTIME数据结构中。可以显式地发送这个消息，也可以利用DateTime_GetSystemtime宏来发送这个消息。

```
DTM_GETSYSTEMTIME
    wParam = 0;
    lParam = (LPARAM) lpSysTime;
```

参数

lpSysTime：指向SYSTEMTIME数据结构的指针。如果DTM_GETSYSTEMTIME返回的是GDT_VALID，那么这个数据结构中将包含系统时间。否则，这个数据结构中将不包含合法信息。这个参数必须是一个合法的指针，不能为NULL。

返回值

如果时间信息被正确地放置在lpSysTime参数中，那么将返回GDT_VALID。如果控件被设置为DTS_SHOWNONE样式并且控件的复选框没有被选中，那么将返回GDT_NONE。如果发生了错误，将返回GDT_ERROR。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DTM_SETFORMAT

根据给定的格式字符串，设置日期与时间检出器控件的显示方式。可以显式地发送这个消息，也可以利用DateTime_SetFormat宏来发送这个消息。

```
DTM_SETFORMAT
    wParam = 0;
    lParam = (LPARAM) lpzFormat;
```

参数

lpzFormat：用来定义所需显示方式，以0结尾的格式字符串format string的地址。如果这个参数被设置为NULL，那么将把控件重新设置为当前样式的缺省格式字符串。

返回值

如果操作成功，则返回非零值；否则，返回0。

说明

为了使得显示的内容更加丰富,建议在格式字符串中加入一些附加的字符。但是,任何不是用于格式处理的字符都必须在格式字符串中用单引号标记包围起来。例如,格式字符串“Today is: 'hh': 'm!': 's' ddddMMMdd, 'yyyy'”将产生这种形式的显示输出:“Today is:04:22:31 Tuesday Mar 23,1996”。

注意 如果DTP控件正在使用缺省格式字符串,那么它将跟踪所发生的改变。如果设置了一个自定义的格式字符串,DTP控件将在响应设置改变时不进行更新。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTM_SETMCCOLOR

为日期与时间检出器控件中的月历部分设置颜色。可以显式发送这个消息,也可以使用DateTime_SetMonthCalColor宏来发送这个消息。

```
DTM_SETMCCOLOR
wParam = (WPARAM)(INT) iColor;
lParam = (LPARAM)(COLORREF) clr;
```

参数

iColor: 一个INT值,用来指定将要设置哪个月历颜色。这个值可以是以下值之一:

MCSC_BACKGROUND	用来设置在月份之间所显示的背景色。
MCSC_MONTHBK	用来设置在月份内部显示的背景色。
MCSC_TEXT	用来设置在月份内部显示的文本颜色。
MCSC_TITLEBK	用来设置在月历标题中显示的背景色。
MCSC_TITLETEXT	用来设置在月历的标题文本中显示的颜色。
MCSC_TRAILINGTEXT	用来设置在第一日文本与最后一日文本中显示的颜色。 第一日与最后一日是指在当前月历中出现的上一个月与下一个月之间的日。

clr: COLORREF值,这个值表示将要在月历的指定区域中显示的颜色。

返回值

如果操作成功,则返回COLORREF值,这个值表示在月历的指定部分中先前的颜色设置;如果操作失败,则返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTM_SETMCFONT

设置将要被日期与时间检出器控件的子月历控件所使用的字体。可以显式地发送这个消息, 也可以利用DateTime_SetMonthCalFont宏来发送这个消息。

```
DTM_SETMCFONT
wParam = (WPARAM) (HFONT) hFont;
lParam = (LPARAM) MAKELONG (fRedraw, 0);
```

参数

hFont: 将要被设置的字体的句柄。

fRedraw: 这个参数用来说明控件是否必须在进行了字体的重新设置之后立刻进行控件的重绘。如果将这个参数设置为TRUE, 将导致控件进行重绘。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTM_SETRANGE

为日期与时间检出器控件设置最大与最小的允许系统时间。可以显式地发送这个消息, 也可以利用DateTime_SetRange宏来发送这个消息。

```
DTM_SETRANGE
wParam = (WPARAM) flags;
lParam = (LPARAM) lpSysTimeArray;
```

参数

flags: 这个参数用来指定哪个值域是合法的。它可以是以下值的组合:

GDTR_MAX 表示SYSTEMTIME数据结构数组中的第二个元素是合法的, 并且它

GDTR_MIN 将被用来设置最大的系统允许时间。
表示SYSTEMTIME数据结构数组中的第一个元素是合法的，并且它将被用来设置最小的系统允许时间。

lpSysTimeArray: SYSTEMTIME数据结构中两个元素数组的地址。SYSTEMTIME数组中的第一个元素包含有最小允许时间，而其中的第二个元素则包含有最大允许时间。不必对在flags参数中没有指定的数组元素进行设置。

返回值

如果操作成功，则返回非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DTM_SETSYSTEMTIME

设置日期与时间检出器控件中的时间。可以显式地发送这个消息，也可以利用DateTime_SetSystemtime宏来发送这个消息。

```
DTM_SETSYSTEMTIME
    wParam = (WPARAM) flag;
    lParam = (LPARAM) lpSysTime;
```

参数

flag: 用来指定将要执行的动作的值。这个值可以是以下值之一：

GDT_NONE 将DTP控件设置为“没有日期项”，并消除它的复选框。如果这个标记被指定，那么lpSysTime将被忽略。需要说明的是，这个标记仅仅作用于DTS_SHOWNONE样式的DTP控件。

GDT_VALID 根据lpSysTime参数中所指向的数据结构中的数据来设置DTP控件。

lpSysTime: SYSTEMTIME数据结构的地址，在这个数据结构中包含有用来设置DTP控件的系统时间。

返回值

如果操作成功，则返回非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

12.4.2 日期与时间检出器控件宏

DateTime_GetMonthCal

获取日期与时间检出器控件中的子月历控件句柄。可以利用这个宏或者通过显式发送DTM_GETMONTHCAL消息的方法来完成这个动作。

```
HWND DateTime_GetMonthCal (  
    HWND hwndDP);
```

参数

hwndDP: DTP控件的句柄。

返回值

DTP控件中子月历控件的句柄。

说明

当用户单击下拉列表箭头时, DTP控件将创建一个子月历控件。如果不再需要这个月历控件, 它将被析构。因此应用程序不能使用DTP控件中子月历控件的静态句柄。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

DTN_CLOSEUP、DTN_DROPDOWN。

DateTime_GetMonthCalColor

获取日期与时间检出器控件中月历控件给定部分的颜色。可以利用这个宏或者通过显式发送DTM_GETMCCOLOR消息方法来完成这个动作。

```
COLORREF DateTime_GetMonthCalColor (  
    HWND hwndDP,  
    int iColor);
```

参数

hwndDP: DTP控件的句柄。

iColor: 一个INT值, 用来说明将要获取哪个月历控件的颜色。这个值可以是以下之一:

MCSC_BACKGROUND	获取月份之间所显示的背景色。
MCSC_MONTHBK	获取月份内部所显示的背景色。
MCSC_TEXT	获取月份中的文本所显示的颜色。
MCSC_TITLEBK	获取月历标题中所显示的背景色。
MCSC_TITLETEXT	获取用来显示月历标题文本的颜色。
MCSC_TRAILINGTEXT	获取用来显示第一日与最后一日文本的颜色。在这里, 第一日与最后一日是指出现在当前月历中上一月份与下一月份的日。

返回值

如果操作成功, 则返回COLORREF值, 这个值用来表示月历中给定部分的颜色设置。否则, 这个宏将返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DateTime_GetMonthCalFont

获取日期与时间检出器控件的子月历控件正在使用的字体。可以利用这个宏或者通过显式发送DTM_GETMCFONT消息的方法来完成这个动作。

```
HFONT DateTime_GetMonthCalFont (
    HWND hwndDP);
```

参数

hwndDP: DTP控件的句柄。

返回值

返回用来表示当前字体句柄的HFONT值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的

Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DateTime_GetRange

返回日期与时间检出器控件的当前最大与最小允许系统时间。可以利用这个宏或者通过显式发送DTM_GETRANGE消息的方法来完成这个动作。

```
DWORD DateTime_GetRange (  
    HWND hwndDP,  
    LPSYSTEMTIME lpSysTimeArray);
```

参数

hwndDP: DTP控件的地址。

lpSysTimeArray: SYSTEMTIME数据结构的两个元素数组地址。

返回值

返回由GDTR_MIN或GDTR_MAX组合而成的DWORD值。SYSTEMTIME数组中的第一个元素包含有最小允许时间, 第二个元素则包含有最大允许时间。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DateTime_GetSystemtime

从日期与时间检出器控件中获取当前选择的时间, 并且将这个时间放置在指定的SYSTEMTIME数据结构中。可以使用这个宏或者通过显式发送DTM_GETSYSTEMTIME消息的方法来完成这个动作。

```
DWORD DateTime_GetSystemtime (  
    HWND hwndDT,  
    LPSYSTEMTIME lpSysTime);
```

参数

hwndDT: DTP控件的句柄。

lpSysTime: 指向SYSTEMTIME数据结构的指针。如果DTM_GETSYSTEMTIME返回GDT_VALID, 那么这个数据结构中将包含系统时间。否则, 这个数据结构中将不包含任何合法信息。这个参数必须是一个合法的指针, 而不能为NULL。

返回值

如果时间信息被成功地放置在lpSysTime参数中,那么将返回GDT_VALID。如果控件被设置为DTS_SHOWNONE样式并且控件的复选框没有被选中,那么将返回GDT_NONE。如果在执行过程中产生了错误,那么将返回GDT_ERROR。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DateTime_SetFormat

根据所给定的格式字符串, 设置日期与时间检出差控件的显示形式。可以使用这个宏或者通过显式发送DTM_SETFORMAT消息的方法来完成这个动作。

```
Void DateTime_SetFormat (
    HWND hwndDT,
    LPCTSTR lpszFormat);
```

hwndDT: DTP控件的句柄。

lpszFormat: 用来定义所需显示方式的以0结尾的format string格式字符串地址。如果将这个参数设置为NULL, 那么将使得控件重新被设置为当前样式的缺省格式字符串形式。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

说明

为了产生更为丰富的显示方式, 建议在格式字符串中包含一些附加的字符。但是, 那些不适用于进行格式化处理的字符必须用单引号括起来。例如, 格式字符串 “'Today is: 'hh':'m':'s ddddMMMd', 'yyyy'” 所产生的输出形式应该是这样的: “Today is:04:22:31 Tuesday Mar 23, 1996”。

注意 当一个DTP控件正在使用缺省格式字符串时, 它将跟踪所发生的改变。如果设置了一个自定义的格式字符串, 控件在对设置改变进行响应时不会进行更新。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DateTime_SetMonthCalColor

为日期与时间检出器控件中的给定月历部分设置颜色。可以使用这个宏或者通过显式发送DTM_SETMCCOLOR消息方法来完成此动作。

```
COLORREF DateTime_SetMonthCalColor (
    HWND    hwndDP,
    int      iColor,
    COLORREF clr);
```

参数

hwndDP: DTP控件的句柄。

iColor: 用来指定将要设置的月历颜色的INT值。这个值可以是以下值之一：

设置值	含义
MCSC_BACKGROUND	设置月份之间所显示的背景色
MCSC_MONTHBK	设置月份内部所显示的背景色
MCSC_TEXT	设置月份中的文本所显示的颜色
MCSC_TITLEBK	设置月历标题中所显示的背景色
MCSC_TITLETEXT	设置用来显示月历标题文本的颜色
MCSC_TRAILINGTEXT	设置用来显示第一日与最后一日文本的颜色。在这里，第一日与最后一日是指出现在当前月历中上一月份与下一月份的日

clr: COLORREF值，这个值用来表示将要为月历控件中特定区域设置的颜色。

返回值

如果操作成功，则返回COLORREF值，这个值表示月历控件中指定部分的先前颜色设置。否则，这个消息将返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

DateTime_SetMonthCalFont

设置将要被日期与时间检出器控件的子月历控件所使用的字体。可以利用这个宏或者通过显式发送DTM_SETMCFONT消息的方法来完成这个动作。

```
void DateTime_SetMonthCalFont (
    HWND hwndDP,
    HFONT hFont,
```



```
LPARAM MAKELONG (fRedraw , 0));
```

参数

hwndDP: DTP控件的句柄。

hFont: 将要被设置的字体的句柄。

fRedraw: 这个参数用来说明在进行字体的设置之后, 是否立刻进行控件的重新绘制。如果将这个参数设置为TRUE, 将导致控件进行重绘。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DateTime_SetRange

为日期与时间检出器控件设置最大与最小允许系统时间。可以利用这个宏或者通过显式发送DTM_SETRANGE消息方法来完成此动作。

```
BOOL DateTime_SetRange (
    HWND hwndDT,
    DWORD flags,
    LPSYSTEMTIME lpSysTimeArray);
```

参数

hwndDT: DTP控件的句柄。

flags: 用来说明哪个范围值是有效的值。这个值可以是以下值的组合:

GDTR_MAX 表示SYSTEMTIME数据结构数组中的第二个元素是合法的, 并且将被用来设置系统的最大允许时间。

GDTR_MIN 表示SYSTEMTIME数据结构数组中的第一个元素是合法的, 并且将被用来设置系统的最小允许时间。

lpSysTimeArray: SYSTEMTIME数据结构中两个元素数组的地址。SYSTEMTIME数组中的第一个元素包含有最小允许时间, 而第二个元素包含有最大允许时间。如果在flags参数中没有进行指定, 就不必为相应的数组元素填充值。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DateTime_SetSystemtime

将一个日期与时间检出器控件分配给定的日期与时间值。可以使用这个宏或者通过显式发送DTM_SETSYSTEMTIME消息的方法来完成这个动作。

```
BOOL DateTime_SetSystemtime (
    HWND hwndDT,
    DWORD flag,
    LPSYSTEMTIME lpSysTime);
```

参数

hwndDT: DTP控件的句柄。

flag: 用来指定将要执行的动作的值。这个参数必须被设置为以下值之一:

GDT_NONE 将DTP控件设置为“没有日期”方式, 并且清除它的复选框。当这个标记被指定时, lpSysTime参数将被忽略。这个标记仅仅对被设置为DTS_SHOWNONE样式的DTP控件有效。

GDT_VALID 根据由lpSysTime所指向的SYSTEMTIME数据结构中的数据来设置DTP控件。

lpSysTime: SYSTEMTIME数据结构的地址, 在这个数据结构中包含将要设置给DTP控件的系统时间信息。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

12.4.3 日期与时间检出器控件通告消息

DTN_CLOSEUP

当用户关闭下拉月历控件时, 日期与时间检出器控件将发送这个通告消息。如果用户从月

历控件中选择了日期或者在月历控件打开时单击了下拉箭头，月历控件将关闭。

```
DTN_CLOSEUP
lpNmhdr = (LPMNHDR) lParam;
```

参数

lpNmhdr: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。

返回值

为这个通告消息所返回的值没有被使用。

说明

DTP控件并不维护一个静态的子月历控件。DTP控件将在发送这个通告消息之前析构这个子月历控件。因此，应用程序不能使用这个控件子月历控件的静态窗口句柄。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 5或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

参见

DTN_DROPDOWN、DTM_GETMONTHCAL。

DTN_DATETIMECHANGE

如果日期与时间检出器控件中发生了任何改变，控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
DTN_DATETIMECHANGE
lpChange = (LPNMDATETIMECHANGE) lParam;
```

参数

lpChange: NMDATETIMECHANGE数据结构的地址，在这个数据结构中包含有关于在控件中所发生改变的信息。

返回值

这个控件的所有者必须返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的

Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTN_DROPDOWN

当用户激活下拉月历时, 日期与时间检出器控件将发送这个通告消息。

DTN_DROPDOWN

lpNmhdr = (LPNMHDR) lParam;

参数

lpNmhdr: NMHDR数据结构的地址, 在这个数据结构中包含关于本通告消息的信息。

返回值

这个通告消息的返回值没有被使用。

说明

通告消息处理程序将要完成的一个任务就是定制下拉月历控件。例如, 如果希望“前进到今天”, 那么就必须设置控件的MCS_NOTODAY样式。如果需要获取月历控件的句柄, 那么就必须向DTP控件发送DTM_GETMONTHCAL消息。然后, 就可以使用这个句柄并且使用SetWindowLong函数来设置所需的月历控件样式。

DTP控件并不维护一个静态子月历控件。DTP控件在发送这个通告消息之前, 将创建一个新的月历控件。此外, 如果这个子月历控件不再活跃, DTP控件将把它析构。因此, 应用程序不能使用DTP控件中子月历控件的静态窗口句柄。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

DTN_CLOSEUP、DTM_GETMONTHCAL。

DTN_FORMAT

在需要请求信号回叫域中显示的文本信息时, 日期与时间检出器控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

DTN_FORMAT

lpNMFormat = (LPNMDATEFORMAT) lParam;

参数

lpNMFormat: NMDATETIMEFORMAT数据结构的地址, 在这个数据结构中包含关于本通告消息实例的信息。而且, 这个数据结构中包含用来定义信号回叫域的子字符串, 并且能够接收控件将要显示的格式字符串。

返回值

控件的所有者必须返回0。

说明

对这个通告消息的处理, 将使得控件的所有者能够提供控件将要显示的自定义字符串(关于信号回叫域的更多信息, 请参见12.1.2节的相关内容)。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTN_FORMATQUERY

在需要获取将要在信号回叫域中显示的字符串的最大允许大小时, 日期与时间检出器控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

DTN_FORMATQUERY

```
lpDTFormatQuery = (LPNMDATETIMEFORMATQUERY) lParam;
```

参数

lpDTFormatQuery: NMDATETIMEFORMATQUERY数据结构的地址, 在这个数据结构中包含有关于信号回叫域的信息。这个数据结构可以包含用来定义信号回叫域的子字符串, 并且可以接收将要在信号回叫域中显示的字符串的最大大小。

返回值

这个控件的所有者必须计算将要在信号回叫域中显示的文本的最大可能宽度, 并且设置NMDATETIMEFORMATQUERY数据结构的szMax数据成员, 然后返回0。

说明

对这个通告消息的处理, 能够使得控件调整将要在特定的信号回叫域中显示的字符串的大小, 从而保证控件在任何时候都能够正确地显示输出信息, 减少控件的显示冲突问题。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者, 使用带有Internet Explorer 3.0或更新版本的

Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTN_USERSTRING

当用户完成在日期与时间检出器控件中进行字符串编辑的动作之后, 日期与时间检出器控件将发送这个通告消息。这个通告消息只能由被设置为DTS_APPCANPARSE样式的DTP控件发送, 并且它是以WM_NOTIFY消息的形式进行发送的。

```
DTN_USERSTRING
    lpDTstring = (LPNMDATETIMESTRING) lParam;
```

参数

lpDTstring: NMDATETIMESTRING数据结构的地址, 在这个数据结构中包含有关于本通告消息实例的信息。

返回值

这个控件的所有者将必须返回0。

说明

这个通告消息的处理, 允许控件所有者为用户在这个控件中所输入的字符串提供自定义的响应方式。控件所有者必须能够对输入字符串进行解析并且在必要时执行适当的动作。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

DTN_WMKEYDOWN

当用户在信号回叫域中输入信息时, 日期与时间检出器控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
DTN_WMKEYDOWN
    lpDTKeystroke = (LPNMDATETIMEWMKEYDOWN) lParam;
```

参数

lpDTKeystroke: NMDATETIMEKEYDOWN数据结构的地址, 在这个数据结构中包含关于本通告消息实例的信息。这个数据结构可以包含用户所输入的键盘信息、用来定义信号回叫域的子字符串以及当前系统日期与时间。

返回值

控件的所有者必须返回0。

说明

对这个通告消息的处理，将允许控件所有者提供对控件中信号回叫域上所发生的击键动作的响应。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

NM_KILLFOCUS (date time)

这个通告消息用来告知日期与时间检出器控件的父窗口，本控件已经失去了输入焦点。

NM_KILLFOCUS通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_KILLFOCUS
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh：NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

所得到的返回值被忽略。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

NM_SETFOCUS (date time)

这个通告消息用来告知日期与时间检出器控件父窗口，控件已经获取了输入焦点。

NM_SETFOCUS通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_SETFOCUS
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh：NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

所得到的返回值被忽略。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

12.4.4 日期与时间检出器控件数据结构

NMDATETIMECHANGE

在这个数据结构中包含有发生在日期与时间检出器控件中的改变信息。这个数据结构被DTN_DATETIMECHANGE通告消息所使用。

```
typedef struct tagNMDATETIMECHANGE {
    NMHDR      nmhdr;
    DWORD      dwFlags;
    SYSTEMTIME st;
} NMDATETIMECHANGE, FAR *LPNMDATETIMECHANGE;
```

成员

nmhdr: 包含关于本通告消息的信息的NMHDR数据结构。

dwFlags: 这个值用来说明日期与时间检出器控件是否被设置为“没有日期”状态（仅仅对于DTS_SHOWNONE样式有效）。这个数据成员标记还能够说明st数据成员的内容是否合法，并且包含有当前时间信息。这个数据成员可以是以下值之一：

GDT_NONE 控件被设置为“没有日期”状态。“没有日期”状态仅仅适用于被设置为DTS_SHOWNONE样式的日期与时间检出器控件。

GDT_VALID 控件没有被设置为“没有日期”状态。其中的数据成员st包含有当前日期与时间信息。

st: 包含关于当前系统中日期与时间信息的SYSTEMTIME数据结构。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMDATETIMEFORMAT

在这个数据结构中，包含有日期与时间检出器控件内定义信号回叫域的格式字符串部分。

这个数据结构拥有用来定义信号回叫域的子字符串，并且包含有一个缓冲器，在这个缓冲器中存放将要在信号回叫域中显示的字符串。这个数据结构被DTN_FORMAT通告消息所使用。

```
typedef struct tagNMDATETIMEFORMAT{
    NMHDR nmhdr;
    LPCTSTR pszFormat;
    SYSTEMTIME st;
    LPCTSTR pszDisplay;
    TCHAR szDisplay [64 ];
}NMDATETIMEFORMAT, FAR *LPNMDATETIMEFORMAT;
```

成员

nmhdr: 包含关于本通告消息的信息的NMHDR数据结构。

pszFormat: 用来定义DTP控件信号回叫域的子字符串的地址。这个子字符串由一个或多个后面跟由NULL字符的“X”字符所组成。

st: 包含将要被格式化的日期与时间的SYSTEMTIME数据结构。

pszDisplay: 一个以null结尾的字符串地址，这个字符串包含有控件的显示文本信息。缺省情况下，这个地址是本数据结构的szDisplay数据成员。

建议将pszDisplay数据成员指向一个现有的字符串。在这种情况下，就不需要为szDisplay设置值。但是，直到另一个DTN_FORMAT通告消息被发送之前或者在控件被析构之前，pszDisplay所指向的字符串必须一直保持有效。

szDisplay: 一个64字符的缓冲器，这个缓冲器可以接收将要在DTP控件中显示的以0结尾的字符串。在使用过程中，没有必要填满整个缓冲器。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

NMDATETIMEFORMATQUERY

在这个数据结构中包含有关于日期与时间检出器控件信号回叫域的信息。事实上，它包含有一个定义信号回叫域的子字符串（从控件的格式字符串中获取）。这个数据结构能够接收将要在信号回叫域中显示的最大数目的文本，并且它是由DTN_FORMATQUERY通告消息所使用的。

```
typedef struct tagNMDATETIMEFORMATQUERY {
    NMHDR nmhdr;
    LPCTSTR pszFormat;
    SIZE szMax;
} NMDATETIMEFORMATQUERY, FAR *LPNMDATETIMEFORMATQUERY;
```

成员

nmhdr: 包含关于本通告消息的信息的NMHDR数据结构。

pszFormat: 用来定义DTP控件信号回调域的子字符串地址。这个子字符串是由后面跟有一个NULL字符的一个或多个“X”字符所组成的。

szMax: 这个数据成员是一个SIZE数据结构，此数据结构必须用要在信号回调域中显示的文本的最大大小进行填充。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMDATETIMESTRING

在这个数据结构中包含有与某个特定编辑操作相关的信息, 这个编辑操作是在日期与时间检出器控件中发生的。这个数据结构可以被DTN_USERSTRING通告消息所使用。

```
typedef struct tagNMDATETIMESTRING{
    NMHDR          nmhdr;
    LPCTSTR        pszUserString;
    SYSTEMTIME      st;
    DWORD          dwFlags;
}NMDATETIMESTRING, FAR *LPNMDATETIMESTRING;
```

成员

nmhdr: 包含关于本通告消息的信息的NMHDR数据结构。

pszUserString: 用户所输入的以0结尾的字符串地址。

st: 在处理DTN_USERSTRING通告消息时, 必须被控件所有者所填充的SYSTEMTIME数据结构。

dwFlags: 信号返回域, 对这个数据成员设置为GDT_VALID将意味着st数据成员是合法的, 而如果将这个数据成员设置为GDT_NONE, 则意味着将要吧控件设置为“没有日期”状态 (仅仅对于DTS_SHOWNONE控件样式有效)。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMDATEIMEWMKEYDOWN

这个数据结构中包含有用来描述以及处理DTN_WMKEYDOWN通告消息的信息。

```
typedef struct tagNMDATEIMEWMKEYDOWN {
    NMHDR        nmhdr;
    int           nVirtKey;
    LPCTSTR       pszFormat;
    SYSTEMTIME    st;
} NMDATEIMEWMKEYDOWN, FAR *LPNMDATEIMEWMKEYDOWN;
```

成员

nmhdr: 包含关于本通告消息的信息的NMHDR数据结构。

nVirtKey: 用来表明用户已经按下了一个键的虚拟键。

pszFormat: 一个以0结尾的子字符串, 这个字符串来自格式字符串, 用来定义信号回叫域。这个只字符串是有后面跟随有一个NULL字符的一个或多个“X”字符所组成的。

st: 包含有来自DTP控件的当前日期与时间信息的SYSTEMTIME数据结构, 这个控件的所有者必须根据用户的输入来修改时间信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

第13章 拖放列表框

拖放列表框 (drag list box) 是一个特殊的列表框, 它允许用户将项目从一个位置拖放到另一个位置。应用程序可以利用拖放列表框来按照某个特定的顺序显示字符串, 并且允许用户通过拖放项目位置的方法来改变字符串的显示顺序。

13.1 使用拖放列表框

拖放列表框具有与窗口相同的样式, 并且处理与标准列表框相同的消息。如果需要创建拖放列表框, 首先应该创建一个标准列表框, 然后调用MakeDragList函数。如果需要将对话框中的一个列表框转换为拖放列表框, 就应该在WM_INITDIALOG消息被处理时调用MakeDragList函数。

13.1.1 拖放列表框消息

拖放列表框通过向父窗口发送拖放列表消息的方法, 告知拖放列表框父窗口有关的拖放事件, 父窗口必须能够处理这些拖放列表框消息。

在MakeDragList函数被调用时, 拖放列表框将注册这个消息。为了获取拖放列表框消息的消息标识符 (数字值), 应该调用RegisterWindowMessage函数并且指定DRAGLISTMSGSTRING值。

拖放列表框消息的wParam参数是拖放列表框的控件标识符, 而lParam参数则是DRAGLISTINFO数据结构的地址, 其中包含有拖放事件通告消息代码以及其他信息。这个消息的返回值取决于通告消息。

13.1.2 拖放列表框通告消息

拖放列表通告消息, 由包含在拖放列表DRAGLISTINFO数据结构的uNotification数据成员标识, 这些通告消息可以是DL_BEGINDRAG、DL_DRAGGING、DL_CANCELDRAG或DL_DROPPED通告消息。

当光标正在列表项目之上并且用户单击了鼠标左键时, 将发送DL_BEGINDRAG通告消息。控件的父窗口可以返回TRUE, 从而开始拖放操作; 或者返回FALSE, 从而不允许进行拖放动作。这样, 父窗口就可以允许某些列表项目进行拖放, 而另一些项目列表则不能进行拖放。利用LBIItemFromPt函数, 可以决定哪个列表项目放置在指定的位置。

如果拖放正在进行, 那么在鼠标移动的任何时候都可以发送DL_DRAGGING通告消息, 也可以在鼠标没有移动时按照一定的间隔时间发送这个通告消息。父窗口首先应该使用LBIItemFromPt数据函数来决定光标所在的列表项目, 然后利用DrawInsert函数绘制插入的图标。如果是LBIItemFromPt函数的bAutoScroll参数指定值为TRUE, 那么就可以在光标位于客户区的上方或下方时使得列表框滚动一行。为这个通告消息所返回的值可以说明鼠标指针的类型, 这个类型是由拖放列表框设置的。

如果用户单击鼠标右键取消了一个拖放操作，或者按下了ESC键，那么将发送一个DL_CANCELDRAG通告消息。如果用户释放鼠标左键，那么即使是光标没有在某一个列表项目上，也将会完成拖放操作，从而发送DL_DROPPED通告消息。拖放列表框在发送这两个通告消息之前首先将释放鼠标捕获信息，这两个通告消息的返回值都将被忽略。

13.2 拖放列表框参考

13.2.1 拖放列表框函数

DrawInsert

在指定拖放列表框的父窗口中绘制所插入的图标。

```
void DrawInsert(
    HWND handParent,
    HWND hLB,
    int nItem
);
```

参数

handParent: 拖放列表框父窗口的句柄。

hLB: 拖放列表框的句柄。

nItem: 将要绘制的图标项目的标识符。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

LBItemFromPt

获取列表框中指定位置的项目的索引。

```
int LBItemFromPt(
    HWND hLB,
    POINT pt,
    BOOL bAutoScroll
);
```

参数

hLB: 将要检查的列表框的句柄。

pt: 包含将要检查的屏幕坐标的POINT数据结构。

bAutoScroll: 屏幕滚动标志位。如果这个参数为TRUE, 并且坐标点正好位于列表框的上方或下方, 那么函数将把列表框滚动一行并返回-1。否则, 函数将不滚动列表框。

返回值

如果坐标点正好位于列表项目的上方, 将返回项目的标识符, 否则返回-1。

说明

只有在自从上次滚动列表框以来经历了很少的时间, **LBItemFromPt**函数才滚动列表框。这种计时方式使得列表框不能在函数反复被调用时过快地滚动列表框, 例如, 当**DL_DRAGGING**通告消息或**WM_MOUSEMOVE**消息正在被处理时就会发生函数反复被调用的情况。

如果指定的点位于列表框客户区的外面, 并且**bAutoScroll**被设置为TRUE, 那么这个函数将滚动列表框, 而不会返回项目标识符。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在**commctrl.h**中进行了声明。

导入库: **comctl32.lib**。

MakeDragList

改变所指定的拖放列表框的单选列表框。

```
BOOL MakeDragList(  
    HWND hLB  
);
```

参数

hLB: 单选列表框的句柄。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在**commctrl.h**中进行了声明。

导入库: **comctl32.lib**。

13.2.2 拖放列表框通告消息

DL_BEGINDRAG

这个通告消息用来告知拖放列表框的父窗口, 用户在某个项目上单击了鼠标左键。拖放列

表框将以拖放列表消息的形式发送DL_BEGINDRAG通告消息。

```
DL_BEGINDRAG
    idCtl = (WPARAM) (int) wParam;
    pDragInfo = (LPARAM) (LPDRAGLISTINFO) lParam;
```

参数

idCtl: 拖放列表框的控件标识符。

pDragInfo: DRAGLISTINFO数据结构的地址, 在这个数据结构中包含有DL_BEGINDRAG通告消息代码、拖放列表框句柄以及光标位置信息。

返回值

如果需要开始进行拖放操作, 则返回TRUE; 否则, 将返回FALSE, 从而阻止拖放操作。

说明

在处理这个通告消息时, 窗口过程一般使用LBItemFromPt函数来决定指定光标位置的列表项目。然后, 根据这个项目是否可以利用被拖动信息, 来返回TRUE或FALSE。在返回TRUE之前, 窗口过程应该保存列表项目的索引, 从而使得应用程序知道哪个项目将要被移动或拷贝(在拖放操作完成之后)。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

DL_CANCELDRAG

这个通告消息用来说明用户已经单击了鼠标右键或按下了ESC键, 从而取消了拖放操作。拖放列表框将以拖放列表消息的形式向父窗口发送DL_CANCELDRAG通告消息。

```
DL_CANCELDRAG
    idCtl = (WPARAM) (int) wParam;
    pDragInfo = (LPARAM) (LPDRAGLISTINFO) lParam;
```

参数

idCtl: 拖放列表框的控件。

pDragInfo: DRAGLISTINFO数据结构的地址, 在这个数据结构中包含有关于DL_CANCELDRAG通告消息代码、拖放列表框句柄以及光标位置的信息。

返回值

没有返回值。

说明

通过处理DL_CANCELDRAG通告消息, 应用程序可以将它的内部状态重新设置为表示拖放操作没有被使用的状态。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

DL_DRAGGING

这个通告消息可以用来说明用户在拖放某个项目的过程中移动了鼠标。即使是鼠标没有移动,也将会在拖放过程中周期性地发送DL_DRAGGING通告消息。拖放列表框是以拖放列表消息的形式向父窗口发送这个通告消息的。

```
DL_DRAGGING
idCtl = (WPARAM) (int) wParam;
pDragInfo = (LPARAM) (LPDRAGLISTINFO) lParam;
```

参数

idCtl: 拖放列表框的控件标识符。

pDragInfo: DRAGLISTINFO数据结构的地址,在这个数据结构中包含有关于DL_DRAGGING通告消息代码、拖放列表框句柄以及光标位置的信息。

返回值

所得到的返回值可以用来判断拖放列表框应该被设置的鼠标光标类型。这个返回值可以是DL_STOPCURSOR、DL_COPYCURSOR或DL_MOVECURSOR。如果返回的是其他值,那么表示光标不会发生改变。

说明

典型情况下,窗口过程将通过判断位于光标下面的项目然后绘制插入图标的方法来处理DL_DRAGGING通告消息。如果需要获取光标下的项目,应该使用LBItemFromPt函数,并且将bAutoScroll参数设置为TRUE。这个选项设置将使得当光标在拖放列表框客户区的上方或下方时周期性地滚动拖放列表框。如果需要绘制插入图标,就应该使用DrawInsert函数。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

DL_DROPPED

这个通告消息用来说明用户释放鼠标左键,从而完成了拖放操作。拖放列表框将以拖放列表消息的形式向父窗口发送DL_DROPPED通告消息。

```
DL_BEGINDRAG
idCtl = (WPARAM) (int) wParam;
pDragInfo = (LPARAM) (LPDRAGLISTINFO) lParam;
```


参数

idCtl: 拖放列表框的控件标识符。

pDragInfo: DRAGLISTINFO数据结构的地址, 在这个数据结构中包含有关于DL_DROPPED通告消息代码、拖放列表框句柄以及光标位置的信息。

返回值

没有返回值。

说明

对这个通告消息的处理, 通常是通过把正在被拖放的项目插入到光标下面、项目之前的列表中来实现的。如果需要获取光标位置上项目的索引, 应该使用LBItemFromPt函数, 需要说明的是, 即使光标没有在列表项目之上, DL_DROPPED通告消息也将被发送。但在这种情况下, LBItemFromPt将返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

13.2.3 拖放列表框数据结构**DRAGLISTINFO**

在这个数据结构中包含有关于拖放事件的信息。指向DRAGLISTINFO数据结构的指针将作为拖放列表消息的IParam参数进行传递。

```
typedef struct {
    UINT uNotification;
    HWND hWnd;
    POINT ptCursor;
} DRAGLISTINFO, FAR *LPDRAGLISTINFO;
```

成员

uNotification: 这个数据成员是用来指明拖放事件类型的通告消息代码, 它可以是以下值之一:

- | | |
|---------------|----------------------------------|
| DL_BEGINDRAG | 表示用户在某个列表项目上单击了鼠标左键。 |
| DL_CANCELDRAG | 表示用户单击了鼠标右键或者按下了ESC键, 从而取消了拖放操作。 |
| DL_DRAGGING | 表示用户在拖放某个项目的过程中移动了鼠标。 |
| DL_DROPPED | 表示用户释放鼠标左键, 从而完成了拖放操作。 |
| hWnd: | 拖放列表框的句柄。 |

ptCursor: POINT数据结构, 在这个数据结构中包含有关于鼠标光标当前x坐标与y坐标位置的信息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。



第14章 平面滚动条

14.1 平面滚动条

Microsoft Internet Explorer 4.0版本中引入了一个名为平面滚动条的可视化技术。从功能上来看,平面滚动条的行为与普通的滚动条一样,唯一的不同之处在于平面滚动条不显示三维形式。

图14-1显示了一个包含平面滚动条的窗口。

注意 平面滚动条API在Comctl32.dll的4.71版本以及后续版本中得以实现。



图14-1 平面滚动条

使用平面滚动条

本节将讨论如何在应用程序中实现平面滚动条。

1. 预先准备工作

为了使用平面滚动条API,必须在应用程序的源代码中包含Commctrl.h头文件,并且将源文件与Comctl32.lib库进行链接。

2. 向窗口中添加平面滚动条

如果需要向一个窗口中添加平面滚动条,那么应该调用InitializeFlatSB函数,并且将句柄传递给窗口。用户不能使用标准的滚动条API来处理这些滚动条,而应该使用FlatSB_xxxx版本的滚动条API。在这个版本中,存在可用来设置与获取滚动信息、滚动范围与位置的平面滚动条API。如果窗口没有对平面滚动条进行初始化,那么平面滚动条API将使用相应的标准API(若存在)。这样就允许用户在没有编写条件代码的情况下来打开或关闭平面滚动条。

由于应用程序可能会为它的平面滚动条设置量度,因此在系统的量度发生改变时,这些被设置的量度不会自动地更新。如果系统滚动条的量度发生改变,那么将广播一个WM_SETTINGCHANGE消息,在这个消息中wParam被设置为SPI_SETNONCLIENTMETRICS。如果需要将平面滚动条更新为使用新的系统量度,那么应用程序就必须处理这个消息,并且改变平面滚动条中与度量相关的属性。

为了改变滚动条属性,应该使用FlatSB_SetScrollProp函数。以下代码段能够将一个平面滚动条中与度量信息相关的属性改变为当前的系统值。

```
void FlatSB_UpdateMetrics(HWND hWnd)
{
    FlatSB_SetScrollProp(hWnd,WSB_PROP_CXVSCROLL,
        GetSystemMetrics(SM_CXVSCROLL),FALSE);
    FlatSB_SetScrollProp(hWnd,WSB_PROP_CXHSCROLL,
```

```

GetSystemMetrics(SM_CXHSCROLL), FALSE);
FlatSB_SetScrollProp(hWnd, WSB_PROP_CYVSCROLL,
GetSystemMetrics(SM_CYVSCROLL), FALSE);
FlatSB_SetScrollProp(hWnd, WSB_PROP_CXHSCROLL,
GetSystemMetrics(SM_CXHSCROLL), FALSE);
FlatSB_SetScrollProp(hWnd, WSB_PROP_CXHTHUMB,
GetSystemMetrics(SM_CXHTHUMB), FALSE);
FlatSB_SetScrollProp(hWnd, WSB_PROP_CYVTHUMB,
GetSystemMetrics(SM_CYVTHUMB), TRUE);
}

```

3. 加强平面滚动条功能

FlatSB_SetScrollProp函数可以用来修改平面滚动条，从而定制窗口外观。可以对竖直滚动条的宽度以及水平滚动条的高度进行设置，也可以设置滚动条方向箭头的宽度（对于水平滚动条）以及高度（对于竖直滚动条）。

FlatSB_SetScrollProp函数还可以用来定制平面滚动条的显示。通过改变WSB_PROP_VSTYLE属性与WSB_PROP_HSTYLE属性，可以设置自己需要使用的滚动条类型。有三种滚动条样式可供使用：

FSB_ENCARTA_MODE

在这种情况下，将显示标准的平面滚动条。当鼠标移动到一个方向按钮或者缩略图上时，滚动条的这个部分将以三维的形式显示出来。

FSB_FLAT_MODE

在这种情况下，将显示标准的平面滚动条。当鼠标移动到一个方向按钮或者缩略图上时，滚动条的这个部分将以反色的方式显示出来。

FSB_REGULAR_MODE

在这种情况下，将显示一个很普通的、非平面滚动条，而且没有任何可视化效果被应用。

4. 删除平面滚动条

如果希望从窗口中删除平面滚动条，就应该调用UninitializeFlatSB API函数，并且将这个句柄传递给窗口。这个API函数将只能在运行状态下从窗口中删除平面滚动条。当窗口被析构时，不必调用这个API函数。

14.2 平面滚动条参考

平面滚动条函数

InitializeFlatSB

为某个特定的窗口初始化平面滚动条。

```

BOOL InitializeFlatSB(
    HWND hwnd
);

```

参数

hwnd: 将要接收平面滚动条的窗口的句柄。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

说明

这个API函数必须在任何其他平面滚动条API被调用之前调用, 缺省情况下, 窗口将接受平面滚动条。可以利用FlatSB_SetScrollProp API函数来改变滚动条的样式。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

FlatSB_EnableScrollBar

打开或关闭平面滚动条的一个或两个方向按钮。如果平面滚动条没有为这个窗口初始化, 那么函数将调用标准的EnableScrollBar API函数。

```
BOOL FlatSB_EnableScrollBar (
    HWND hwnd,
    int wSBflags,
    UINT wArrows
);
```

参数

hwnd: 包含这个平面滚动条的窗口的句柄。这个窗口句柄必须在先前调用InitializeFlatSB函数时被传递。

wSBflags: 用来指定滚动条类型的参数, 这个参数可以是下列值之一:

SB_BOTH 打开或关闭滚动条水平方向与竖直方向上的方向按钮。

SB_HORZ 打开或关闭滚动条水平方向上的方向按钮。

SB_VERT 打开或关闭滚动条竖直方向上的方向按钮。

wArrows: 用来说明滚动条箭头的打开状态以及说明哪个箭头被打开或关闭的参数。这个参数可以是下列值时以一:

ESB_DISABLE_BOTH 关闭指定滚动条两个方向按钮。

ESB_DISABLE_DOWN 关闭竖直滚动条的向下方向按钮。

ESB_DISABLE_LEFT 关闭水平滚动条的向左方向按钮。

ESB_DISABLE_LTUP 关闭水平滚动条的向左方向按钮, 或者关闭竖直滚动条的向

	上方向按钮。
ESB_DISABLE_RIGHT	关闭水平滚动条的向右方向按钮。
ESB_DISABLE_RTDN	关闭水平滚动条的向右方向按钮，或者关闭竖直滚动条的向下方向按钮。
ESB_DISABLE_UP	关闭竖直滚动条的向上方向按钮。
ESB_ENABLE_BOTH	打开指定滚动条中两个方向上的方向按钮。

返回值

如果滚动条发生了改变，则返回一个非零值，否则返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

FlatSB_GetScrollInfo

获取平面滚动条的信息。如果平面滚动条没有被窗口初始化，那么这个函数将调用标准的GetScrollInfo API函数。

```

BOOL FlatSB_GetScrollInfo (
    HWND hwnd,
    int fnBar,
    LPSCROLLINFO lpsi
);

```

参数

hwnd: 包含平面滚动条的窗口句柄。在先前对InitializeFlatSB函数的调用时，这个窗口句柄必须被传递给这个函数。

fnBar: 用来指定滚动条类型的参数，这个参数可以是下列值之一：

SB_HORZ 获取水平滚动条的信息。

SB_VERT 获取竖直滚动条的信息。

lpsi: SCROLLINFO数据结构的地址，这个数据结构将接收指定滚动条的信息。在调用FlatSB_GetScrollInfo函数之前，这个数据结构中的cbSize数据成员与fMask数据成员必须已经被赋值。其中，fMask数据成员用来指明应该获取哪些属性信息，它可以是以下值的组合：

SIF_PAGE 获取平面滚动条的页面信息。所获取的信息将放置在SCROLLINFO数据结构中的nPage数据成员中。

SIF_POS 获取平面滚动条的位置信息。所获取的信息将放置在SCROLLINFO数

据结构的nPos数据成员中。

SIF_RANGE 获取平面滚动条的范围信息。所获取的信息将放置在SCROLLINFO数据结构结构的nMin数据成员与nMax数据成员中。

SIF_ALL SIF_PAGE、SIF_POS与SIF_RANGE值的组合。

返回值

如果操作成功，则返回非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

导入库：comctl32.lib。

FlatSB_GetScrollPos

获取平面滚动条中的缩略图位置。如果平面滚动条没有为窗口进行初始化，那么这个函数将调用标准的GetScrollPos API函数。

```
int FlatSB_GetScrollPos (
    HWND hwnd,
    int code
);
```

参数

hwnd：包含平面滚动条的窗口的句柄。这个窗口句柄必须在先前对InitializeFlatSB函数的调用过程中被传递给这个函数。

code：用来指定滚动条类型的参数，这个参数必须是下列值之一：

SB_HORZ 获取水平滚动条的缩略图位置信息。

SB_VERT 获取竖直滚动条的缩略图位置信息。

返回值

返回指定平面滚动条当前缩略图位置。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

导入库：comctl32.lib。

FlatSB_GetScrollProp

获取平面滚动条的属性。这个函数也可以用来判断窗口是否调用过InitializeFlatSB函数。

```
BOOL FlatSB_GetScrollProp (
    HWND hwnd,
    UNIT index,
    LPINT pValue
);
```

参数

hwnd：包含平面滚动条的窗口句柄。这个窗口句柄必须在先前的InitializeFlatSB函数调用中被传递给这个函数。

index：这个参数可用来决定pValue所代表的内容以及哪个属性将要被获取，它的值可以是以下之一：

WSB_PROP_CXHSCROLL

表示pValue是一个INT值的地址，这个INT值可用来获取水平滚动条方向按钮的宽度，单位为像素。

WSB_PROP_CXHTHUMB

表示pValue是一个INT值的地址，这个INT值可用来获取水平滚动条缩略图的宽度，单位为像素。

WSB_PROP_CXVSCROLL

表示pValue是一个INT值的地址，这个INT值可用来获取垂直滚动条方向按钮的宽度，单位为像素。

WSB_PROP_CYHSCROLL

表示pValue是一个INT值的地址，这个INT值可用来获取水平滚动条方向按钮的高度，单位为像素。

WSB_PROP_CYVSCROLL

表示pValue是一个INT值的地址，这个INT值可用来获取垂直滚动条方向按钮的高度，单位为像素。

WSB_PROP_CYVTHUMB

表示pValue是一个INT值的地址，这个INT值可用来获取垂直滚动条缩略图的高度，单位为像素。

WSB_PROP_HBKGCOLOR

表示pValue是一个COLORREF值的地址，这个值可用来接收水平滚动条的背景色。

WSB_PROP_HSTYLE

表示pValue是一个INT值的地址，这个INT值可用来获取水平滚动条的下列视觉效果之一：

FSB_ENCARTA_MODE

这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时，滚动条的这个部分将以三维的方式显示出来。

FSB_FLAT_MODE

这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时，滚动条的这个部分将以反色的方式显示出来。

WSB_PROP_PALETTE**FSB_REGULAR_MODE**

这时将显示一个普通的非平面滚动条。在这种方式下没有任何特殊的视觉效果。

表示pValue是一个HPALETTE值的地址，这个值可用来接收在绘制滚动条时所使用的调色板。

WSB_PROP_VBKGCOLOR

表示pValue是一个COLORREF值的地址，这个值可用来接收竖直滚动条的背景色。

WSB_PROP_VSTYLE

表示pValue是一个INT值的地址，这个值可用来接收关于竖直滚动条的下列视觉效果之一：

FSB_ENCARTA_MODE

这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时，滚动条的这个部分将以三维的方式显示出来。

FSB_FLAT_MODE

这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时，滚动条的这个部分将以反色的方式显示出来。

FSB_REGULAR_MODE

这时将显示一个普通的非平面滚动条。在这种方式下没有任何特殊的视觉效果。

WSB_PROP_WINSTYLE

表示pValue是一个INT值的地址，这个值可用来获取当前窗口所使用的WS_HSCROLL与WS_VSCROLL样式位。

pValue：用来获取所需数据的地址，这个参数的值取决于在index中所传递的标志位。

返回值

如果操作成功，则返回非零值；否则，返回0。如果index的值是WSB_PROP_HSTYLE，那么如果InitializeFlatSB函数已经被这个窗口所调用，就将返回非零值，否则返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

导入库：comctl32.lib。

FlatSB_GetScrollRange

获取平面滚动条的滚动范围。如果平面滚动条没有被窗口初始化，那么这个函数将调用标

准的GetScrollRange API函数。

```
BOOL FlatSB_GetScrollRange (
    HWND hwnd,
    int code,
    LPINT lpMinPos,
    LPINT lpMaxPos
);
```

参数

hwnd: 包含这个平面滚动条的窗口句柄。这个窗口句柄必须在先前的InitializeFlatSB函数调用中被传递给这个函数。

code: 用来说明滚动条类型的参数, 这个参数可以是下列值之一:

SB_HORZ 表示需要获取水平滚动条的滚动范围。

SB_VERT 表示需要获取竖直滚动条的滚动范围。

lpMinPos: INT值的地址, 可用来获取最小滚动范围值。

lpMaxPos: INT值的地址, 可用来获取最大滚动范围值。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

FlatSB_SetScrollInfo

为平面滚动条设置信息。如果平面滚动条没有被窗口所初始化, 那么这个函数将调用标准的SetScrollInfo API函数。

```
int FlatSB_SetScrollInfo (
    HWND hwnd,
    int fnBar,
    LPSCROLLINFO lpsi,
    BOOL fRedraw
);
```

参数

hwnd: 包含平面滚动条的窗口句柄。这个窗口句柄必须在先前对InitializeFlatSB函数的调用过程中被传递给这个函数。

fnBar: 用来指定滚动条类型参数, 这个参数可以是以下值之一:

SB_HORZ 用来设置水平滚动条的信息。

SB_VERT 用来设置竖直滚动条的信息。

lpsi: **SCROLLINFO**数据结构的地址, 在这个数据结构中包含有关于指定滚动条的新信息。在调用**FlatSB_SetScrollInfo**函数之前, 这个数据结构的**cbSize**数据成员与**fMask**数据成员必须已经被赋值。其中, **fMask**数据成员用来说明数据结构中的哪些数据成员包含有合法信息, 它可以是以下值的组合:

SIF_DISABLENOSCROLL 如果新信息将会导致滚动条被删除, 那么就关闭滚动条。
SIF_ALL 这个值是由**SIF_PAGE**、**SIF_POS**与**SIF_RANGE**组合成的。

SIF_PAGE 用来设置平面滚动条的页面信息。**nPage**数据结构中的**SCROLLINFO**数据成员必须包含新页面值。

SIF_POS 用来为平面滚动条设置位置信息。**SCROLLINFO**数据结构的**nPos**数据成员必须包含新位置值。

SIF_RANGE 用来设置平面滚动条的范围信息。**SCROLLINFO**数据结构的**nMin**数据成员与**nMax**数据成员必须包含新范围值。

fRedraw: 用来说明滚动条是否需要在发生改变之后立刻进行重绘的参数。如果这个参数的值为**TRUE**, 那么滚动条将进行重绘; 如果这个参数值为**FALSE**, 那么滚动条将不进行重绘。

返回值

返回当前滚动条位置。如果对**FlatSB_SetScrollInfo**函数的调用改变了滚动条位置, 那么所返回的是先前位置。

环境需求

需要使用动态链接库**Comctl32.dll**的4.71版本或更新版本。

Windows NT/2000: 需要使用**Windows 2000** (或者, 使用带有**Internet Explorer 4.0**或更新版本的**Windows NT 4.0**)。

Windows 95/98: 需要使用**Windows 98** (或者, 使用带有**Internet Explorer 4.0**或更新版本的**Windows 95**)。

Windows CE: 不支持**Windows CE**。

头文件: 在**comctl.h**中进行了声明。

导入库: **comctl32.lib**。

FlatSB_SetScrollPos

设置平面滚动条中缩略图的当前位置。如果平面滚动条没有为窗口进行初始化, 那么函数将调用标准的**SetScrollInfo** API函数。

```
int FlatSB_SetScrollPos (
    HWND hwnd,
    int code,
    int nPos
```

```

    BOOL fRedraw
);

```

参数

hwnd: 包含平面滚动条的窗口句柄。这个窗口句柄必须在先前对InitializeFlatSB函数的调用过程中被传递给这个函数。

code: 用来指定滚动条类型的参数, 这个参数可以是下列值之一:

SB_HORZ 设置水平滚动条的缩略图位置。

SB_VERT 设置竖直滚动条的缩略图位置。

nPos: 用来指定新缩略图位置的参数。

fRedraw: 用来说明滚动条是否需要在发生改变之后立刻进行重绘的参数。如果这个参数的值为TRUE, 那么滚动条将进行重绘; 如果这个参数值为FALSE, 那么滚动条将不进行重绘。

返回值

返回指定平面滚动条中缩略图的先前位置。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

FlatSB_SetScrollProp

为平面滚动条设置属性。

```

BOOL FlatSB_SetScrollProp (
    HWND hwnd,
    UINT index,
    int newValue
    BOOL fRedraw
);

```

参数

hwnd: 包含平面滚动条的窗口句柄。这个窗口句柄必须在先前对InitializeFlatSB函数的调用过程中被传递给这个函数。

index: 这个参数用来决定newValue所表示的内容, 以及哪个属性将被设置。它可以是下列值之一:

WSB_PROP_CXHSCROLL

表示newValue是一个INT值, 这个INT值可用来表示水平滚动条方向按钮的宽度, 单位为像素。

- WSB_PROP_CXHThumb** 表示new Value是一个INT值, 这个INT值可用来表示水平滚动条缩略图的宽度, 单位为像素。
- WSB_PROP_CXVScroll** 表示new Value是一个INT值, 这个INT值可用来表示竖直滚动条方向按钮的宽度, 单位为像素。
- WSB_PROP_CYHScroll** 表示new Value是一个INT值, 这个INT值可用来表示水平滚动条方向按钮的高度, 单位为像素。
- WSB_PROP_CYVScroll** 表示new Value是一个INT值, 这个INT值可用来表示竖直滚动条方向按钮的高度, 单位为像素。
- WSB_PROP_CyVThumb** 表示new Value是一个INT值, 这个INT值可用来表示竖直滚动条缩略图的高度, 单位为像素。
- WSB_PROP_HBkgColor** 表示new Value是一个COLORREF值, 这个值可用来表示水平滚动条的背景色。
- WSB_PROP_HStyle** 表示new Value是一个可用来改变水平滚动条下列视觉效果的值之一:
FSB_ENCARTA_MODE
 这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时, 滚动条的这个部分将以三维的方式显示出来。
FSB_FLAT_MODE
 这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时, 滚动条的这个部分将以反色的方式显示出来。
FSB_REGULAR_MODE
 这时将显示一个普通的非平面滚动条。在这种方式下没有任何特殊的视觉效果。
- WSB_PROP_PALETTE** 表示new Value是一个HPALETTE值, 用来表示在绘制滚动条时所使用的调色板。
- WSB_PROP_VBkgColor** 表示new Value是一个COLORREF值, 这个值可用来表示竖直滚动条的背景色。
- WSB_PROP_VStyle** 表示new Value是一个可用来改变关于竖直滚动条的下列视觉效果的值之一:
FSB_ENCARTA_MODE
 这时将显示一个标准的平面滚动条。当鼠标移到某个方向按钮或者缩略图上时, 滚动条的这个部分将以三维的方式显示出来。
FSB_FLAT_MODE
 这时将显示一个标准的平面滚动条。当鼠标移到某个方

向按钮或者缩略图上时，滚动条的这个部分将以反色的方式显示出来。

FSB_REGULAR_MODE

这时将显示一个普通的非平面滚动条。在这种方式下没有任何特殊的视觉效果。

newValue: 将要被设置的新值，这个参数取决于在index中所传递的标志位。

fRedraw: 用来说明滚动条是否需要在发生改变之后立刻进行重绘的参数。如果这个参数的值为TRUE，那么滚动条将进行重绘；如果这个参数值为FALSE，那么滚动条将不进行重绘。

返回值

如果操作成功，则返回非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件: 在comctl.h中进行了声明。

导入库: comctl32.lib。

FlatSB_SetScrollRange

设置平面滚动条的滚动范围。如果平面滚动条没有被窗口进行初始化，那么这个函数将调用标准的SetScrollRange API函数。

```
int FlatSB_SetScrollRange(
    HWND hwnd,
    int code,
    int nMinPos,
    int nMaxPos,
    BOOL fRedraw
);
```

参数

hwnd: 包含平面滚动条的窗口句柄。这个窗口句柄必须在先前对InitializeFlatSB函数的调用过程中被传递给这个函数。

code: 用来说明滚动条类型的参数，这个参数可以是下列值之一：

SB_HORZ 设置水平滚动条的滚动范围。

SB_VERT 设置竖直滚动条的滚动范围。

nMinPos: 用来指定新的最小滚动范围值的参数。

nMaxPos: 用来指定新的最大滚动范围值的参数。

fRedraw: 用来说明滚动条是否需要在发生改变之后立刻进行重绘的参数。如果这个参数的值为TRUE, 那么滚动条将进行重绘; 如果这个参数值为FALSE, 那么滚动条将不进行重绘。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

FlatSB_ShowScrollBar

用来控制显示或隐藏平面滚动条。如果平面滚动条没有被窗口进行初始化, 那么这个函数将调用标准的ShowScrollBar API函数。

```
BOOL FlatSB_ShowScrollBar (
    HWND hwnd,
    int code
    BOOL fShow
);
```

参数

hwnd: 包含平面滚动条的窗口句柄。这个窗口句柄必须在先前对InitializeFlatSB函数的调用过程中被传递给这个函数。

code: 用来说明滚动条类型的参数, 这个参数可以是下列值之一:

SB_BOTH 显示或隐藏水平滚动条与竖直滚动条。

SB_HORZ 显示或隐藏水平滚动条。

SB_VERT 显示或隐藏竖直滚动条。

fShow: 用来指定滚动条应该显示还是隐藏的参数。如果这个参数非零, 那么滚动条将被显示; 而如果这个参数为0, 那么滚动条将被隐藏。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

UninitializeFlatSB

为某个特定的窗口取消对平面滚动条的初始化。这样, 指定的窗口将使用标准的滚动条。

HRESULT UninitializeFlatSB (

HWND hwnd

);

参数

hwnd: 将要被取消初始化的平面滚动条的窗口句柄。

返回值

返回下列值之一:

E_FAIL 窗口中有一个滚动条当前处于使用状态。这时, 函数的操作将不能正常完成。

S_FALSE 窗口目前还没有对平面滚动条进行初始化。

S_OK 函数操作成功。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。



第15章 头标控件

头标控件 (header control) 是一个窗口, 它通常位于文本或数值列的上方。在头标控件中, 为每个列设置有一个标题, 并且头标窗口可以被划分为几个部分。用户可以拖动用来分开头标控件中不同部分的分隔符, 从而为每个列设置宽度。图15-1显示了一个头标控件, 在这个头标控件中的各个标题上给出了关于目录中文件的详细信息。

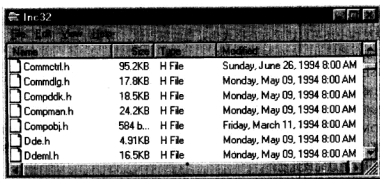


图15-1 头标控件

15.1 使用头标控件

可以利用CreateWindowEx函数来创建头标控件, 并且在使用函数的过程中指定WC_HEADER窗口类与适当的头标样式。在进行通用控件动态链接库 (DLL) 的加载时, 这个窗口类将被注册。为了确信DLL确实被加载, 应该使用InitCommonControlsEX函数。在创建头标控件之后, 就应该考虑将它划分为几个部分, 为每个部分设置文本, 并且利用头标窗口消息来控制窗口的外观。

15.1.1 头标控件大小与位置

典型情况下, 必须将头标控件的大小与位置设置在某个特定矩形的边界区域之内, 例如, 在窗口的客户区内。利用HDM_LAYOUT消息, 可以从头标控件中获取适当的大小与位置值。

在发送HDM_LAYOUT消息时, 应指定HDLAYOUT数据结构的地址, 其中这个数据结构包含有头文件将要放置在其中的矩形的坐标以及一个指向WINDOWPOS数据结构的指针。头标控件将利用适合于指定矩形顶部的位置与大小值来填充WINDOWPOS数据结构。它的高度值是控件的水平边界高度与当前选中到控件的设备上下文中字符平均高度之和。

如果希望使用HDM_LAYOUT来设置初始大小以及头标控件的位置, 那么就应该将控件的初

始可视状态设置为隐藏状态。在发送HDM_LAYOUT来获取大小与位置值之后,就可以使用SetWindowPos函数来设置新的大小、位置与可视状态。

15.1.2 项目

典型情况下,一个头标控件都有几个用来定义控件中列的头标项目。可以向头标控件中发送HDM_INSERTITEM消息,从而在头标控件中添加一个项目,在所发送的消息中,包含有HDITEM数据结构的地址。这个数据结构用来定义头标项目的属性,这个头标项目可以包括一个字符串、一个位图图像、一个初始大小以及一个由应用程序所定义的32位值。

项目HDITEM数据结构中的fmt数据成员可以包括HDF_STRING或HDF_BITMAP标记值,用来指示控件是否需要显示项目的字符串或位图。如果希望项目既显示一个字符串又显示一个位图,那么就应该将fmt数据成员设置为包括HDF_OWNERDRAW标志位,来创建自绘项目。在HDITEM数据结构中还可以指定进行格式化处理的标志位,用来告知控件是否需要在项目的矩形中将字符串或位图居中、左对齐或右对齐。

HDM_INSERTITEM将返回新添加的项目的索引。可以在其他消息中使用这个索引来设置关于这个项目的属性或者获取关于这个项目的信息。可以用HDM_DELETEITEM消息来删除某个项目,这时必须在这个消息中指定将要删除的项目索引。

可以用HDM_SETITEM消息来设置某个现有头标项目的属性,用HDM_GETITEM消息来获取某个项目的当前属性。如果需要获取某个头标控件中的项目数,就应该使用HDM_GETITEMCOUNT消息完成此动作。

15.1.3 自绘头标控件

可以定义某个头标控件中的单独项目为自绘项目。这项技术能够提供对头标项目外观显示的更多控制方式。

HDM_INSERTITEM消息可用来将一个新的自绘项目插入到头标控件中,而HDM_SETITEM消息则可以用来将某个现有项目改变为自绘项目。在这两个消息中都包含有HDITEM数据结构的地址,在这个数据结构中的fmt数据成员必须被设置为HDF_OWNERDRAW值。

当某个头标控件必须绘制自绘项目时,它将向父窗口发送WM_DRAWITEM消息,这个消息中的wParam参数应该是头标控件的子窗口标识符,而lParam参数应该是DRAWITEMSTRUCT数据结构的地址。父窗口将使用这个数据结构中的信息用来对项目进行绘制。对于头标控件中的某个自绘项目,DRAWITEMSTRUCT数据结构中包含下列信息:

成 员	描 述
CtlType	这个数据成员表示ODT_HEADER自绘控件类型
CtlID	这个数据成员表示头标控件的子窗口标识符
itemID	将要被绘制的项目的索引
itemAction	这个数据成员表示ODA_DRAWENTIRE绘制动作标志位
itemState	如果光标正在这个项目之上并且鼠标按钮被按下,那么这个数据成员表示的是ODS_SELECTED自绘动作标志位。否则,这个数据成员为0

(续)

成 员	描 述
hwndItem	这个数据成员表示头标控件的句柄
hDC	这个数据成员表示头标控件设备上下文的句柄
rcItem	这个数据成员表示将要被绘制的头标项目的坐标。这里所指的坐标是相对于头标控件左上角的相对坐标
itemData	这个数据成员表示于项目相关的由应用程序定义的32位值

15.1.4 头标控件通告消息

当用户单击或双击某个项目、拖放某个控件分隔标记以及改变项目属性时，头标控件将向它的父窗口发送通告消息。父窗口将以WM_NOTIFY消息的形式接收这个通告消息。头标控件可以使用下列通告消息：

通告消息	描 述
HDN_BEGINTRACK	这个通告消息用来说明将要开始进行分隔标记的拖动
HDN_DIVIDERDBLCLICK	这个通告消息用来说明用户双击分隔标记
HDN_ENDTRACK	这个通告消息用来说明分隔标记拖动动作的结束
HDN_ITEMCHANGED	这个通告消息用来说明项目的属性发生了改变
HDN_ITEMCHANGING	这个通告消息用来说明项目的属性将要发生改变
HDN_ITEMCLICK	这个通告消息用来说明用户单击了某个项目
HDN_ITEMDBLCLICK	这个通告消息用来说明用户双击了某个项目
HDH_TRACK	这个通告消息用来说明用户拖动了某个分隔标记

15.1.5 缺省头标控件消息处理过程

本节将描述窗口过程为WC_HEADER窗口类所处理的窗口消息。

消 息	所执行的处理动作
WM_CREATE	初始化头标控件
WM_DESTROY	取消初始化头标控件
WM_ERASEBKGD	利用控件的当前背景色来填充头标控件的背景
WM_GETDLGCODE	返回DLGC_WANTTAB值与DLGC_WANTARROWS值的组合
WM_GETFONT	返回当前字体的句柄，这个字体被头标控件用来绘制控件的文本
WM_LBUTTONDOWNBLCLK	捕获鼠标输入动作。如果鼠标位于一个分割标记上，控件将发送HDN_BEGINTRACK通告消息，并开始拖放分隔标记。如果鼠标位于某个项目上，那么这个项目将显示为被按下的状态
WM_LBUTTONDOWN	这个消息与WM_LBUTTONDOWNBLCLK消息相同
WM_LBUTTONUP	释放鼠标捕获的信息。如果控件正在跟踪鼠标运动，那么控件将发送HDN_ENDTRACK通告消息并且重绘头标控件。否则，控件将发送HDN_ITEMCLICK通告消息并且重绘被单击的头标项目
WM_MOUSEMOVE	如果某个分割标记正在被拖动，那么控件将发送HDN_TRACK通告消息并且新的位置显示被拖动的项目。如果鼠标左键被按下并且鼠标正位于某个项目上，那么这个项目将显示按下状态

(续)

消 息	所执行的处理动作
WM_NCCREATE	分配并初始化某个内部数据结构
WM_NCDESTROY	在头标控件被取消初始化之后, 将释放由头标控件所分配的资源
WM_PAINT	绘制头标控件中的非法区域。如果wParam参数不是NULL, 那么控件将假设这个参数是HDC, 并且利用这个设备上下文进行绘制
WM_SETCURSOR	设置鼠标的形状, 这个形状取决于鼠标是在分割标记上还是在头标项目上
WM_SETFONT	为头标控件中的设备上下文选择一个新的字体句柄

15.1.6 创建头标控件

下面的例子将演示如何创建头标控件, 并且在父窗口的客户区顶部放置这个头标控件。在初始时, 头标控件是隐藏的。HDM_LAYOUT消息用来计算控件大小以及它在父窗口中的位置。然后, 控件将被重新定位并且被显示出来:

```
// DoCreateHeader -creates a header control that is
//      positioned along the top of the parent window's
//      client area.
// Returns the handle to the header control.
// hwndParent -handle to the parent window.
//
// Global variable
//      g_hinst -handle to the application instance
extern HINSTANCE g_hinst;

HWND DoCreateHeader(HWND hwndParent)
{
    HWND hwndHeader;
    RECT rcParent;
    HDLAYOUT hdl;
    WINDOWPOS wp;

    // Ensure that the common control DLL is loaded,
    // and then create the header control.
    InitCommonControlsEx();

    if ((hwndHeader =CreateWindowEx(0,WC_HEADER,(LPCTSTR)NULL,
        WS_CHILD |WS_BORDER |HDS_BUTTONS |HDS_HORZ,
        0,0,0,hwndParent,(HMENU)ID_HEADER,g_hinst,
        (LPVOID)NULL))!=NULL)
        return (HWND)NULL;

    // Retrieve the bounding rectangle of the parent
    // window's client area,and then request size and
    // position values from the header control.
```

```

GetClientRect(hwndParent,&rcParent);

hdl.prc =&rcParent;
hdl.pwpos =&wp;
if (!SendMessage(hwndHeader,HDM_LAYOUT,0,(LPARAM)&hdl))
    return (HWND)NULL;

// Set the size,position,and visibility of the
// header control.
SetWindowPos(hwndHeader,wp.hwndInsertAfter,wp.x,wp.y,
    wp.cx,wp.cy,wp.flags |SWP_SHOWWINDOW);

return hwndHeader;
}

```

15.1.7 向头标控件中添加项目

下面的例子将演示如何使用HDM_INSERTITEM消息以及HDITEM数据结构，将一个项目添加到头标控件中。新的头标项目包括一个在项目的矩形中进行了左对齐的字符串：

```

//DoInsertItem -inserts an item into a header control.
//Returns the index of the new item.
//hwndHeader -handle to the header control.
//iInsertAfter -index of the previous item.
//nWidth -width of the new item.
//lpsz -address of the item string.
int DoInsertItem(HWND hwndHeader,int iInsertAfter,
    int nWidth,LPSTR lpsz)
{
    HDITEM hdi;
    int index;

    hdi.mask =HDI_TEXT |HDI_FORMAT |HDI_WIDTH;
    hdi.pszText =lpsz;
    hdi.cxy =nWidth;
    hdi.cchTextMax =strlen(hdi.pszText);
    hdi.fmt =HDF_LEFT |HDF_STRING;

    index =SendMessage(hwndHeader,HDM_INSERTITEM,
        (LPARAM)iInsertAfter,(LPARAM) &hdi);

    return index;
}

```

15.2 在Internet Explorer中更新头标控件

Microsoft Internet Explorer中的头标控件支持下列新特性。

1. 图像列表

支持这个特性的新消息是HDM_GETIMAGELIST消息与HDM_SETIMAGELIST消息。头标控件的HDITEM数据结构已经进行了更新，从而能够支持图像列表功能。图像列表可用于支持当前位图。

2. 信号回叫项目

目前，信号回叫支持包括头标项目文本与图像的信号回叫。如果将某个头标项目的文本设置为LPSTR_TEXTCALLBACK值，或者将头标项目的图像设置为I_IMAGECALLBACK值，那么将导致这个控件发送HDN_GETDISPINFO消息，从而请求所需要的信号回叫信息。HDN_GETDISPINFO由新的NMHDDISPINFO数据结构来支持。

3. 自定义项目顺序

支持这个特性的新消息包括：HDM_GETORDERARRAY、HDM_SETORDERARRAY与HDM_ORDERTOINDEX消息。

4. 拖放处理

目前，利用拖放的方法对头标项目进行重新排序的功能已经可以使用。在创建控件时，应该包含HDS_DRAGDROP样式来实现拖放支持。缺省情况下，头标控件将自动处理进行拖放动作的覆盖效果与图像效果。但是，控件的所有者可以通过处理HDN_BEGINDRAG通告消息与HDN_ENDDRAG通告消息的方法来手工支持拖放处理动作。在处理这些消息时，控件所有者将可能需要发送HDM_CREATEDRAGIMAGE消息与HDM_SETHOTDIVIDER消息。

5. 热跟踪

当这个特性被打开时，位于鼠标指针下的项目将显示高亮状态。可以调用SystemParametersInfo函数来判断热跟踪功能是否被打开。在一个头标控件中，为了打开热跟踪功能，应该用HDS_HOTTRACK样式来创建这个头标控件。

6. 文本、位图与图像

头标项目可以同时显示项目的文本、位图与图像。

15.3 头标控件样式

头标控件有许多样式，这些样式可决定控件的外观与行为。在创建头标控件时，就可以设置头标控件的初始样式。在创建头标控件之后，为了获取与改变头标控件的样式，应该使用GetWindowLong函数与SetWindowLong函数。

1. HDS_BUTTONS

在这种样式下，头标控件中的每个项目都具有与下按按钮相同的外观与行为。如果在用户单击头标控件中的某个项目时，应用程序需要执行某个任务，那么这个样式就十分有用。例如，应用程序可以根据用户所单击项目的不同来对列中的信息进行排序。

2. HDS_DRAGDROP

这个样式在4.70版本中有效，它允许对头标项目进行拖放方式的重新排序。

3. HDS_FILTERBAR

这个样式在5.80版本中有效，其中包含有一个过滤条，作为标准头标控件的一部分。这个过

滤条允许用户非常方便地在显示时应用一个过滤器。对HDM_LAYOUT函数的调用将导致为头标控件设置新的大小,并使得列表视图得到更新。

4. HDS_FULDRAG

这个样式在4.70版本中有效,它使得即使在用户改变列大小时也能够使头标控件显示列的内容。

5. HDS_HIDDEN

这个样式用来表明将要被隐藏的头标控件。事实上,它不会隐藏头标控件;相反,当向带有HDS_HIDDEN样式的头标控件发送HDM_LAYOUT消息时,这个控件将在WINDOWPOS数据结构中的cy数据成员中返回0。这时,就可以将控件的高度设置为0,从而隐藏这个控件。当用户希望利用这个控件作为信息的容器,而不是作为可视的控件时,这个样式十分有用。

6. HDS_HORZ

这个样式创建一个带有水平方向标记的头标样式。

7. HDS_HOTTRACK

这个样式在4.70版本中有效,用来打开热跟踪功能。

15.4 头标控件参考

15.4.1 头标控件消息

HDM_CLEARFILTER

为一个给定的头标控件清除过滤器。可以显式地发送这个消息,也可以使用Header_ClearFilter宏来发送这个消息。

```
HDM_CLEARFILTER
wParam = (WPARAM)(HWND) hwnd;
lParam = (LPARAM)(int) i;
```

参数

hwnd: 头标控件的句柄。

i: 用来表明将要清除的过滤器所在的列号。

返回值

返回将要被清除的过滤器控件的索引号。

说明

如果列号被指定为-1,那么所有的过滤器将被清除,HDM_FILTERCHANGE通告消息将只被发送一次。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Header_ClearAllFilters。

HDM_CREATEDRAGIMAGE

创建项目图像的半透明版本, 用来作为拖动时所使用的图像。可以显式地发送这个消息, 也可以利用Header_CreaterDragImage宏来发送这个消息。

```
HDM_CREATEDRAGIMAGE
wParam = (WPARAM)(int) iIndex;
lParam = 0;
```

参数

iIndex: 头标控件中项目基于0的索引。被分配给这个项目的图像将作为产生半透明图像的基准。

返回值

返回一个图像列表的句柄, 在这个图像列表中只有这个新的图像一个元素。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_DELETEITEM

从头标控件中删除一个项目。可以显式地发送这个消息, 也可以使用Header_DeleteItem宏来发送这个消息。

```
HDM_DELETEITEM
wParam = (WPARAM)(int) index;
lParam = 0;
```

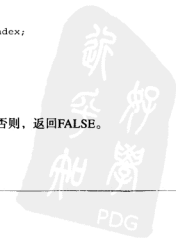
参数

index: 将要被删除的项目索引。

返回值

如果操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求



Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_EDITFILTER

开始编辑指定的过滤器。

```
HDM_EDITFILTER
    wParam = (WPARAM)(int) i;
    lParam = MAKELPARAM(fDiscardChanges, 0)
);
```

参数

i: 用来指定将要被编辑的列号。

fDiscardChanges: 用来指定如何处理用户编辑改变信息的标志位。如果用户正处于编辑过
滤器的过程中, 当这个消息发送时, 这个标记为将指定进行何种动作:

TRUE 放弃用户所做的任何改变。

FALSE 接受用户所做的任何改变。

返回值

返回将要被编辑的过滤器索引。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的
Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的
Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDM_CLEARFILTER。

HDM_GETBITMAPMARGIN

获取头标控件中位图页边空白的宽度。可以显式地发送这个消息, 也可以使用
Header_GetBitmapMargin宏来发送这个消息。

```
HDM_GETBITMAPMARGIN
    wParam = 0;
    lParam = 0;
```

返回值

返回位图页边空白的宽度值, 单位为像素。如果先前没有指定位图的页边空白大小, 那么

将返回位图页面宽度的缺省值3*GetSystemMetrics(CX_EDGE)。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDM_SETBITMAPMARGIN。

HDM_GETIMAGELIST

获取已经为现有头标控件设置的图像列表的句柄。可以显式地发送这个消息, 也可以使用Header_GetImageList宏来发送这个消息。

```
HDM_GETIMAGELIST
    wParam = 0;
    lParam = 0;
```

返回值

返回为头标控件设置的图像列表句柄。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_GETITEM

获取头标控件中某个项目的信息。可以显式地发送这个消息, 也可以使用Header_GetItem宏来发送这个消息。

```
HDM_GETITEM
    wParam = (WPARAM)(int) index;
    lParam = (LPARAM)(LPHDITEM) phdi;
```

参数

index: 将要从中获取信息的项目索引。

phdi: HDITEM数据结构的地址。当这个消息被发送时, mask数据成员将用来说明被请求的

信息类型。这个消息返回之后，其他数据成员就可以获取所请求的信息。如果mask数据成员被指定为0，那么这个消息将返回TRUE，但不向这个数据结构拷贝任何信息。

返回值

如果操作成功，则返回TRUE；否则，返回FALSE。

说明

如果HDITEM数据结构mask数据成员的HDI_TEXT标志位被设置，那么这个数据结构的pszText数据成员将被改变为指向新的文本，而不是利用所请求的文本来填充缓冲器。应用程序不应该假设文本总是被放置在所请求的缓冲器中。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_GETITEMCOUNT

获取头标控件中的项目数。可以显式地发送这个消息，也可以使用Header_GetItemCount宏来发送这个消息。

```
HDM_GETITEMCOUNT
    wParam = 0;
    lParam = 0;
```

返回值

如果操作成功，则返回头标控件中的项目数，否则返回-1。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_GETITEMRECT

为头标控件中的给定项目获取约束矩形。可以显式地发送这个消息，也可以使用Header_GetItemRect宏来发送这个消息。

```
HDM_GETITEMRECT
    wParam = (WPARAM)(int) iIndex;
    lParam = (LPARAM) lpItemRect;
```

参数

iIndex：将要从中获取约束矩形的头标控件项目基于0的索引。

lpItemRect：RECT数据结构的地址，在这个数据结构中包含有约束矩形的信息。

返回值

若操作成功，则返回一个非0值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_GETORDERARRAY

获取头标控件中项目当前从左向右的顺序信息。可以显式地发送这个消息，也可以使用Header_GetOrderArray宏来发送这个消息。

```
HDM_GETORDERARRAY
wParam = (WPARAM)(int) iSize;
lParam = (LPARAM)(LPINT) lpiArray;
```

参数

iSize: lpiArray数组可以容纳的整型元素数目。这个值必须等于控件中的项目数（请参见HDM_GETITEMCOUNT）。

lpiArray: 整型数组的地址，这个数组中包含有头标控件项目的索引值。这个数组的元素数将由iSize参数来指定，而且必须大于或等于控件中的项目数。例如，下列代码段将能够有足够的内存来容纳索引值：

```
int iItems,
    *lpiArray;

//Get memory for buffer.
(iItems =SendMessage(hWndHD,HDM_GETITEMCOUNT,0,0))!=-1)
    if(!lpiArray =calloc(iItems,sizeof(int)))
        MessageBox(hWndd,"Out of memory.", "Error",MB_OK);
```

返回值

如果操作成功，则返回非零值，并且lpiArray数组中的缓冲器将按照头标控件中项目从左到右的出现顺序来获取项目的编号。否则，这个消息将返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_GETUNICODEFORMAT

为头标控件获取UNICODE字符格式标志。可以显式地发送这个消息，也可以使用Header_GetUnicodeFormat宏来发送这个消息。

HDM_GETUNICODEFORMAT

```
wParam = 0;  
lParam = 0;
```

返回值

返回头标控件的UNICODE格式标志值。如果这个值为非零，那么表示控件正在使用UNICODE字符；否则，表示控件正在使用ANSI字符。

说明

关于这个消息的更多讨论，请参见CCM_GETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

HDM_SETUNICODEFORMAT。

HDM_HITTEST

对一个屏幕点进行测试，以便判断在这个指定的点上有哪个头标项目（若存在）。

HDM_HITTEST

```
wParam = 0;  
lParam = (LPARAM)(LPHDHITTESTINFO) phdhti;
```

参数

phdhti：HDHITTESTINFO数据结构的地址，在这个数据结构中包含有将要测试的点的位置信息，并且这个数据结构可以用来获取关于测试结果的信息。

返回值

返回指定点的位置上所存在的项目的索引（若存在），否则返回-1。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_INSERTITEM

将一个新项目插入到头标控件中。可以显式地发送这个消息，也可以使用Header_InsertItem宏来发送这个消息。

```
HDM_INSERTITEM
wParam = (WPARAM)(int) index;
lParam = (LPARAM)(const LPHDITEM) phdi;
```

参数

index: 项目的索引号，在这个索引之后将插入新项目。如果index大于或等于控件中的项目数，那么新项目将插入到头标控件的结尾。如果index为0，那么新项目将插入到头标控件的起始位置。

phdi: HDITEM数据结构的地址，在这个数据结构中包含有关于新项目的信息。

返回值

如果操作成功，则返回新项目的索引，否则返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_LAYOUT

获取头标控件在父窗口中的当前大小与位置信息。可以显式地发送这个消息，也可以使用Header_Layout宏来发送这个消息。

```
HDM_LAYOUT
wParam = 0;
lParam = (LPARAM)(LPHDLAYOUT) pLayout;
```

参数

pLayout: HDLAYOUT数据结构的地址，其中这个数据结构的prc数据成员将指定矩形所在的坐标，pwpos数据成员将获取矩形中头标控件的大小与位置。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_ORDERTOINDEX

根据某个头标项目在头标控件中的顺序，获取这个头标控件的索引。可以显式地发送这个

消息,也可以使用Header_OrderToIndex宏来发送这个消息。

```
HDM_ORDERTOINDEX
wParam = (WPARAM) iOrder;
lParam = 0;
```

参数

iOrder: 项目在头标控件中出现的顺序值(从左向右顺序)。例如,头标控件中最左端列所对应的项目索引值为0,这个项目右端的下一个项目索引就是1,如此等等。

返回值

返回用来说明项目索引的INT值。如果iOrder的值非法(为负或者过大),那么将返回与iOrder相等的值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_SETBITMAPMARGIN

设置一个现有头标控件中位图的页边空白宽度,单位为像素。可以显式地发送这个消息,也可以使用Header_SetBitmapMargin宏来发送这个消息。

```
HDM_SETBITMAPMARGIN
wParam = (WPARAM)(int) iWidth;
lParam = 0;
```

参数

iWidth: 在现有头标控件中,用来包围某个位图的页边空白宽度,单位为像素。

返回值

返回位图页边空白的宽度,单位为像素。如果先前没有指定位图的页边空白宽度,那么将返回缺省值3*GetSystemMetrics(CX_EDGE)。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98(或者,使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDM_GETBITMAPMARGIN。

HDM_SETFILTERCHANGETIMEOUT

设置某个过滤器的属性发生改变的时刻与HDM_FILTERCHANGED通告消息被发送的时刻之间所允许的最大时间间隔（过期时间）。可以显式地发送这个消息，也可以使用Header_SetFilterChangeTimeout宏来发送这个消息。

```
HDM_SETFILTERCHANGETIMEOUT
wParam = 0;
lParam = (LPARAM) i;
```

参数

i: 过期时间值，单位为毫秒。

返回值

返回正在被修改的过滤器控件的索引。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

HDM_FILTERCHANGE。

HDM_SETHOTDIVIDER

改变头标项目之间的分隔标记颜色，从而用来说明某次外部拖放操作的拖放目标。可以显式地发送这个消息，也可以使用Header_SetHotDivider宏来发送这个消息。

```
HDM_SETHOTDIVIDER
wParam = (WPARAM) flag;
lParam = (LPARAM) dwInputValue;
```

参数

flag: 用来说明由dwInputValue所代表的值的类型，这个值可以是以下之一：

TRUE 表示dwInputValue保存有指针的客户区坐标。

FALSE 表示dwInputValue保存有分隔标记的索引值。

dwInputValue: 保存在dwInputValue参数中的值由flag参数的值进行解释。

如果flag参数为TRUE，那么dwInputValue将表示鼠标指针的x坐标与y坐标。其中，x坐标在

低字部分，y坐标在高字部分。当头标控件接收到这个消息时，它将根据dwInputValue坐标的值来高亮显示适当的分隔标记。

如果flag参数为FALSE，那么dwInputValue参数将表示将要被高亮显示的分隔标记的整型索引。

返回值

返回的值等于头标控件所高亮显示的分割标记索引值。

说明

如果头标控件使用HDS_DRAGDROP样式，那么这个消息将能够产生头标控件会自动生成的效果。当头标控件的所有者手工地处理拖放操作时，那么将有可能使用HDM_SETHOTDIVIDER消息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_SETIMAGELIST

为一个现有的头标控件分配图像列表。可以显式地发送这个消息，也可以使用Header_SetImageList宏来发送这个消息。

```
HDM_SETIMAGELIST
wParam = 0;
lParam = (LPARAM) hIm1;
```

参数

hIm1：图像列表的句柄。

返回值

返回先前与这个头标控件相关联的图像列表的句柄。如果操作失败或者先前没有指定任何图像列表，那么将返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_SETITEM

设置头标控件中指定项目的属性。可以显式地发送这个消息，也可以使用Header_SetItem宏来发送这个消息。

```
HDM_SETITEM
    wParam = (WPARAM)(int) iIndex;
    lParam = (LPARAM)(const LPHDITEM) phdItem;
```

参数

iIndex：其属性将要被修改的项目的当前索引。

phdItem：HDITEM数据结构的地址，在这个数据结构中包含有关于项目的信息。这个消息被发送时，这个数据结构的mask数据成员必须被进行了设置，用来说明哪个属性将要被设置。

返回值

若操作成功，则返回一个非0值；否则，返回0。

说明

支持这个消息的HDITEM数据结构也能够支持存放项目的顺序信息与图像列表信息。通过使用这些数据成员，可以控制项目显示的顺序并且指定与项目一同出现的图像。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDM_SETORDERARRAY

按照从左到右的顺序设置头标项目索引。可以显式地发送这个消息，也可以使用Header_SetOrderArray宏来发送这个消息。

```
HDM_SETORDERARRAY
    wParam = (WPARAM)(int) iSize;
    lParam = (LPARAM) lpiArray;
```

参数

iSize：lpiArray数组的缓冲器大小，单位为元素数。这个值必须等于由HDM_GETITEMCOUNT所返回的值。

lpiArray：一个数组的地址，这个数组用来指定项目从左到右显示的次序。例如，如果这个数组中的内容是{2,0,1}，那么控件将从左到右地显示项目2、项目0与项目1。

返回值

若操作成功，则返回一个非0值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版

本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_SETUNICODEFORMAT

为控件设置UNICODE字符格式标志。这个消息允许在运行时改变控件所使用的字符集, 而不必重新创建控件。可以显式地发送这个消息, 也可以使用Header_SetUnicodeFormat宏来发送这个消息。

```
HDM_SETUNICODEFORMAT
wParam = (WPARAM)(BOOL) fUnicode;
lParam = 0;
```

参数

fUnicode: 这个参数用来决定由控件所使用的字符集。如果参数值为非零, 那么控件将使用UNICODE字符。如果这个参数值为0, 那么控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式字符标志。

说明

关于这个消息的更多讨论, 请参见CCM_SETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDM_GETUNICODEFORMAT。

15.4.2 头标控件宏

Header_ClearFilter

清除给定头标控件的过滤器。可以使用这个宏, 也可以通过显式发送HDM_CLEARFILTER消息的方法来完成此动作。

```
int Header_ClearFilter (
    hwnd,
    i
);
```

参数

hwnd: 头标控件的句柄。

i: 用来说明将要被清除的过滤器的列值。如果这个参数值被设置为-1, 那么将清除所有的过滤器。

说明

如果这个列值被指定为-1, 那么所有的过滤器将被清除并且HDN_FILTERCHANGE通告消息将只被发送一次。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Header_ClearAllFilters。

Header_CreateDragImage

创建现有头标控件中项目图标透明版本。可以使用这个宏, 也可以通过显式发送HDM_CREATEDRAGIMAGE消息的方法来完成此动作。

```
HIMAGELIST Header_CreateDragImage (
    HWND hwndHD,
    int iIndex
);
```

参数

hwndHD: 头标控件的句柄。

iIndex: 头标控件中项目的基于0的索引。分配给这个项目的图像将作为创建透明图像时所使用的基准。

返回值

返回图像列表的句柄, 在这个图像列表中只包含一个新的图像作为图像列表的唯一元素。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

Header_DeleteItem

从标题控件中删除一个项目。可以使用这个宏，也可以通过显式发送HDM_DELETEITEM消息的方法来完成此动作。

```
BOOL Header_DeleteItem (
    hwndHD,
    index
);
```

参数

hwndHD：标题控件的句柄。

index：将要被删除的项目的索引。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

Header_DeleteItem宏的定义如下：

```
#define Header_DeleteItem(hwndHD, index) \
    (BOOL) SendMessage((hwndHD), HDM_DELETEITEM, (WPARAM)(int)(index), 0L)
```

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

Header_EditFilter

开始编辑某个指定的过滤器控件。

```
int Header_EditFilter (
    hwnd,
    i,
    fDiscardChanges
);
```

参数

hwnd：标题控件的句柄。

i：这个参数值用来指明将要被编辑的过滤器列。

fDiscardChanges：这个参数用来指明如何处理用户的编辑改变。如果用户正在编辑某个过滤器，当这个消息被发送时，此参数将用来指定完成何种动作。

TRUE：放弃用户所做的任何改变。

FALSE：接受用户所做的任何改变。

返回值

返回正在被编辑的过滤器控件的索引值。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDM_EDITFILTER。

Header_GetBitmapMargin

获取现有头标控件中某个位图的页边空白宽度, 单位为像素。可以使用这个宏, 也可以通过显式发送HDM_GETBITMAPMARGIN消息的方法来完成此动作。

```
Header_GetBitmapMargin (
    hwnd
);
```

参数

hwnd: 头标控件的句柄。

返回值

返回位图页边空白的宽度, 单位为像素。如果先前没有指定位图的页边空白, 那么将返回缺省值3*GetSystemMetrics(CX_EDGE)。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Header_SetBitmapMargin。

Header_GetImageList

获取已经为现有头标控件所设置的图像列表的句柄。可以使用这个宏, 也可以通过显式发

送HDM_GETIMAGELIST消息的方法来完成此动作。

```
HIMAGELIST Header_GetImageList (HWND hwndHD);
```

参数

hwndHD: 头标控件的句柄。

返回值

返回为头标控件所设置的图像列表的句柄。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_GetItem

获取关于头标控件中某个项目的信息。可以使用这个宏, 也可以通过显式发送HDM_GETITEM消息的方法来完成此动作。

```
BOOL Header_GetItem (
    HWND hwndHD,
    int index,
    LPHDITEM phdi
);
```

参数

hwndHD: 头标控件的句柄。

index: 将要为其获取信息的项目的索引。

phdi: HDITEM数据结构的地址。当这个消息被发送时, mask数据成员将指定被请求的信息类型。消息返回之后, 数据结构中的其他数据成员将接受所请求的信息。如果mask数据成员被指定为0, 那么消息将返回TRUE, 但不拷贝任何信息到数据结构中。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

说明

如果HDITEM数据结构mask数据成员的HDI_TEXT标记被设置了值, 那么控件将把数据结构的pszText数据成员改变为指向新的文本, 而不是利用所请求的文本来填充缓冲器。应用程序不应该假设文本总是被放置在所请求的缓冲器中。

Header_GetItem宏的定义如下:

```
#define Header_GetItem(hwndHD, index, phdi) \
    (BOOL) SendMessage ((hwndHD), HDM_GETITEM, \
```

```
(WPARAM)(int)(index), (LPARAM)(LPHDITEM)(phdi))
```

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_GetItemCount

获取头标控件中的项目数。可以使用这个宏, 也可以通过显式发送HDM_GETITEMCOUNT消息的方法来完成此动作。

```
int Header_GetItemCount (
    hwndHD
);
```

参数

hwndHD: 头标控件的句柄。

返回值

如果操作成功, 则返回项目数; 否则, 返回-1。

说明

Header_GetItemCount宏的定义如下:

```
#define Header_GetItemCount(hwndHD) \
    (int)SendMessage((hwndHD),HDM_GETITEMCOUNT, 0,0L)
```

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_GetItemRect

获取头标控件中给定项目的约束矩形。可以使用这个宏, 也可以通过显式发送HDM_GETITEMRECT消息的方法来完成此动作。

```
BOOL Header_GetItemRect (
    HWND hwndHD,
    int iIndex,
    LPRECT lpItemRect
);
```

参数

hwndHD: 头标控件的句柄。

iIndex: 将要从中获取约束矩形的头标控件项目基于0的索引。

lpItemRect: RECT数据结构的地址, 这个数据结构可以接收关于约束矩形的信息。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_GetOrderArray

获取头标控件中项目从左到右的当前顺序。可以使用这个宏, 也可以通过显式发送HDM_GETORDERARRAY消息的方法来完成此动作。

```
BOOL Header_GetOrderArray (
    HWND hwndHD,
    int iSize,
    int *lpiArray
);
```

参数

hwndHD: 头标控件的句柄。

iSize: lpiArray数组能够容纳的整型元素数。这个值必须大于或等于控件中的项目数 (请参见HDM_GETITEMCOUNT)。

lpiArray: 一个整型数组的地址, 这个数组可以接收头标控件中项目的索引值。数组中的元素数由iSize参数来指定, 并且数组的元素数必须大于或等于控件中的项目数。例如, 下列代码段将能够保留足够的内存用来存放索引值:

```
int iItems,
    *lpiArray;

//Get memory for buffer
if(! (iItems = SendMessage(hwndHD, HDM_GETITEMCOUNT, 0, 0)) != -1)
    if(! (lpiArray = calloc(iItems, sizeof(int))))
        MessageBox(hwnd, "Out of memory.", "Error", MB_OK);
```

返回值

如果操作成功, 则返回非零值, 并且lpiArray数组中的缓冲器将按照头标控件中项目从左到右的出现顺序获取每个项目的项目编号。否则, 将返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_GetUnicodeFormat

获取控件的UNICODE字符格式标志。可以使用这个宏, 也可以通过显式发送HDM_GETUNICODEFORMAT消息的方法来完成此动作。

```
BOOL Header_GetUnicodeFormat (  
    HWND hwnd  
);
```

参数

hwnd: 控件的句柄。

返回值

返回控件UNICODE格式标记。如果返回值为非零, 那么表示控件正在使用UNICODE字符; 否则, 如果返回值为0, 那么表示控件正在使用ANSI字符。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Header_SetUnicodeFormat。

Header_InsertItem

将一个新的项目插入头标控件中。可以使用这个宏, 也可以通过显式发送HDM_INSERTITEM消息的方法来完成此动作。

```
int Header_InsertItem (  
    HWND hwndHD,  
    int index,  
    phdi  
);
```

参数

hwndHD: 头标控件的句柄。

index: 项目的索引, 在这个索引值之后将插入新的项目。如果index参数值大于或等于控件

中的项目数,那么新项目将插入到头标控件的尾部。如果index为0,那么新项目将插入到头标控件的开始处。

phdi: HDITEM数据结构的地址,在这个数据结构中包含有关于新项目的信息。

返回值

如果操作成功,则返回新项目的索引;否则返回-1。

说明

Header_InsertItem宏的定义如下:

```
#define Header_InsertItem(hwndHd, index, phdi) \
    (int) SendMessage((hwndHd), HDM_INSERTITEM, (WPARAM)(int)(index), \
        (LPARAM)(const LPHDITEM)(phdi))
```

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_Layout

获取父窗口中头标控件的正确大小与位置。可以使用这个宏,也可以通过显式发送HDM_LAYOUT消息的方法来完成此动作。

```
BOOL Header_Layout (
    hwndHd,
    payout
);
```

参数

hwndHd: 头标控件的句柄。

payout: HDLAYOUT数据结构的地址。这个数据结构中的prc数据成员用来指定矩形的坐标, pwpos数据成员用来获取矩形中头标控件的大小与位置。

返回值

若操作成功,则返回TRUE;否则,返回FALSE。

说明

Header_Layout宏的定义如下:

```
#define Header_Layout(hwndHd, payout) \
    (BOOL) SendMessage((hwndHd), HDM_LAYOUT, 0, \
        (LPARAM)(PLHDLayout)(payout))
```

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

Header_OrderToIndex

根据项目在头标控件中的顺序来获取项目的索引值。可以使用这个宏，也可以通过显式发送HDM_ORDERTOINDEX消息的方法来完成此动作。

```
int Header_OrderToIndex (  
    HWND hwndHD,  
    int iOrder  
);
```

参数

hwndHD：头标控件的句柄。

iOrder：项目在头标控件中从左到右的顺序。头标控件中位于最左端的项目的索引值为0，这个项目右边项目的索引值则为1，如此等等。

返回值

返回用来说明项目索引的INT值。如果iOrder参数非法（为负或者过大），那么返回值将与iOrder参数相等。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

Header_SetBitmapMargin

为一个现有头标控件中的位图设置页边空白宽度。可以使用这个宏，也可以通过显式发送HDM_SETBITMAPMARGIN消息的方法来完成此动作。

```
int Header_SetBitmapMargin (  
    hwnd,  
    iWidth  
);
```

参数

hwnd：头标控件的句柄。

iWidth：包含在现有头标控件中位图周围的页边空白的宽度，单位为像素。

返回值

返回位图页边空白的宽度，单位为像素。如果先前没有设定为页边空白，那么将返回缺省值3*GetSystemMetrics(CX_EDGE)。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Header_GetBitmapMargin。

Header_SetFilterChangeTimeout

设置过滤器属性发生改变的时刻与HDN_FILTERCHANGED通告消息被发送的时刻之间的最大时间间隔(过期时间)。可以使用这个宏, 也可以通过发送HDM_SETFILTERCHANGETIMEOUT消息的方法来完成此动作。

```
int Header_SetFilterChangeTimeout (  
    hwnd  
    i  
);
```

参数

hwnd: 头标控件的句柄。

i: 过期时间值, 单位为毫秒。

返回值

返回正在被修改的过滤器控件的索引。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDN_FILTERCHANGE、HDM_SETFILTERCHANGETIMEOUT。

Header_SetHotDivider

改变头标项目时间的分隔标记的颜色, 用来表示外部拖放操作的拖放目标。可以使用这个宏, 也可以通过发送HDM_SETHOTDIVIDER消息的方法来完成此动作。

```
int Header_SetHotDivider (
    HWND hwndHD,
    BOOL flag,
    DWORD dwInputValue
);
```

参数

hwndHD: 头标控件的句柄。

flag: 用来指定dwInputValue参数将如何被解释的值, 这个值可以是以下之一:

TRUE 表示dwInputValue参数包含的是鼠标指针的客户区坐标。

FALSE 表示dwInputValue参数包含的是分隔标记的索引值。

dwInputValue: 保存在dwInputValue参数中的值, 由flag参数的值进行解释。

如果flag参数为TRUE, 那么dwInputValue将表示鼠标指针的x坐标与y坐标。其中, x坐标在低字部分, y坐标在高字部分。当头标控件接收到这个消息时, 它将根据dwInputValue坐标的值来高亮显示适当的分隔标记。

如果flag参数为FALSE, 那么dwInputValue参数将表示将要被高亮显示的分隔标记的整型索引。

返回值

返回被控件高亮显示的分隔标记索引。

说明

被设置为HDS_DRAGDROP样式的头标控件将自动产生这个效果。当控件的所有者需要手工地处理拖放操作时, 将使用这个消息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_SetImageList

将一个图像列表分配给现有头标控件。可以使用这个宏, 也可以通过显式发送HDM_SETIMAGELIST消息的方法来完成此动作。

```
HIMAGELIST Header_SetImageList (
    HWND hwndHD,
    HIMAGELIST hIml
);
```

参数

hwndHD: 头标控件的句柄。

himgl: 图像列表的句柄。

返回值

返回先前被分配给这个头标控件的图像列表句柄, 如果先前没有指定图像列表, 那么将返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_SetItem

设置在头标控件中指定项目的属性。可以使用这个宏, 也可以通过显式发送HDM_SETITEM消息的方法来完成此动作。

```
BOOL Header_SetItem (
    hwndHD,
    iIndex,
    phdItem
);
```

参数

hwndHD: 头标控件的句柄。

iIndex: 其属性将要改变的项目的当前索引。

phdItem: HDITEM数据结构的地址, 在这个数据结构中包含有关于项目的信息。当这个消息被发送时, 数据结构的mask数据成员必须被设置, 从而指示哪个属性将被设置。

返回值

若操作成功, 则返回一个非0值; 否则, 返回0。

说明

支持这个宏的HDITEM数据结构也支持项目顺序信息与图像列表信息。通过使用这些数据成员, 可以控制项目被显示的顺序以及指定与项目一同出现的图像。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_SetOrderArray

为头标项目设置从左到右的顺序。可以使用这个宏，也可以通过显式发送HDM_SETORDERARRAY消息的方法来完成此动作。

```
BOOL Header_SetOrderArray (
    HWND hwndHD,
    int iSize
    int *lpiArray
);
```

参数

hwndHD: 头标控件的句柄。

iSize: lpiArray数组缓冲器的大小，单位为元素数。这个值必须等于由HDM_GETITEMCOUNT所返回的值。

lpiArray: 一个数组的地址，这个数组将指出项目从左到右显示的顺序，例如，如果这个数组中的内容是{2,0,1}，那么控件将从左到右分别显示项目2、项目0与项目1。

返回值

若操作成功，则返回一个非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

Header_SetUnicodeFormat

为控件设置UNICODE字符格式标志。这个消息允许在运行时改变由控件所使用的字符集，而不必重新创建控件。可以使用这个宏，也可以通过显式发送HDM_SETUNICODEFORMAT消息的方法来完成此动作。

```
BOOL Header_SetUnicodeFormat (
    HWND hwnd,
    BOOL fUnicode
);
```

参数

hwnd: 控件的句柄。

fUnicode: 这个参数可以决定由控件所使用的字符集。如果这个参数值为非零，那么控件将使用UNICODE字符。如果这个参数值为0，那么控件将使用ANSI字符。

返回值

返回控件先前的UNICODE格式标志。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

Header_GetUnicodeFormat。

15.4.3 头标控件通告消息

HDN_BEGINDRAG

当在头标控件的项目上开始进行了拖动操作时, 头标控件将发送这个通告消息。这个通告消息只能由被设置为HDS_DRAGDROP样式的头标控件发送, 它是以WM_NOTIFY消息的形式进行发送的。

```
HDN_BEGINDRAG
    pNMHeader = (LPNMHEADER) lParam;
```

参数

pNMHeader: NMHEADER数据结构的地址, 在这个数据结构中包含有关于正在被拖动的头标项目的信息。

返回值

如果需要允许头标控件自动管理拖放操作, 则返回FALSE。如果头标控件的所有者自己执行项目的拖放重排序操作, 则返回TRUE。

说明

缺省情况下, 头标控件将自动地管理拖放重排序操作。如果返回TRUE, 那么表示允许进行外部拖放管理, 从而使得控件所有者能够将被拖放操作的重排序服务作为定制服务的一部分。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_BEGINTRACK

这个通告消息将告知头标控件的父窗口, 用户已经在头标控件中开始拖动了分隔标记 (也

就是说,在鼠标光标位于头标控件的分隔标记上时,用户已经按下了鼠标左键)。这个通告消息是以WM_NOTIFY消息形式进行发送的。

```
HDN_BEGINTRACK
    phdn = (LPMHEADER) lParam;
```

参数

phdn: NMHEADER数据结构的地址,在这个数据结构中包含有关于头标控件以及将要被拖动的分隔标记项目的信息。

返回值

如果允许需要对分隔标记进行跟踪,则返回FALSE;否则,返回TRUE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_DIVIDERDBLCLICK

这个通告消息用来告知头标控件的父窗口,用户已经双击了控件的分隔标记区域。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
HDN_DIVIDERDBLCLICK
    phdn = (LPMHEADER) lParam;
```

参数

phdn: NMHEADER数据结构的地址,在这个数据结构中包含有关于头标控件与已经被双击的分隔标记项目的信息。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_ENDDRAG

当完成了头标控件中项目的拖放操作之后,头标控件将发送这个通告消息,这个通告消息是以WM_NOTIFY消息的形式进行发送的。只有在头标控件被设置为HDS_DRAGDROP样式时,控件才发送这个通告消息。

```
HDN_ENDDRAG
    pNMHeader = (LPMHEADER) lParam;
```

参数

pNMHeader: NMHEADER数据结构的地址, 在这个数据结构中包含有关于正在被拖动的头标项目的信息。

返回值

如果允许控件自动地放置项目以及重新对项目排序, 那么就返回FALSE。如果需要阻止项目被放置, 那么就应该返回TRUE。

说明

如果控件的所有者正在执行外部拖放管理任务, 那么将返回FALSE。然后, 控件所有者将必须通过发送HDM_SETITEM与HDM_SETORDERARRAY通告消息的方法来对头标项目进行重排序。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_ENDTRACK

这个通告消息用来告知头标控件的父窗口, 用户完成了对分隔标记的拖放操作。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
HDM_ENDTRACK
phdn = (LPNMHEADER) lParam;
```

参数

phdn: NMHEADER数据结构的地址, 在这个数据结构中包含有关于头标控件以及其分隔标记已经被拖动的项目的信息。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDM_FILTERBTNCLICK

当过滤器控件被单击或者在响应HDM_SETITEM消息时, 将使用这个通告消息来告知头标

控件的父窗口。

```
HDN_FILTERBTNCLICK
    phdr = (LPNMHDFILTERBTNCLICK) lParam;
```

参数

phdr: NMHDFILTERBTNCLICK数据结构的地址, 在这个数据结构中包含有关于头标控件以及头标过滤器按钮的信息。

返回值

如果返回TRUE, 那么将向头标控件父窗口发送一个HDN_FILTERCHANGED通告消息, 这个通告消息为父窗口提供了一次与用户界面元素进行同步的机会。如果不希望这个通告消息被发送, 则返回FALSE。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

NMHDFILTERBTNCLICK。

HDN_FILTERCHANGE

这个通告消息用来告知头标控件的父窗口, 头标控件过滤器的属性正在被改变或被编辑。

```
HDN_FILTERCHANGE
    phdr = (LPNMHEADER) lParam;
```

参数

phdr: NMHEADER数据结构的地址, 在这个数据结构中包含有关于头标控件与头标项目的信息 (这些信息包括将要发生改变的属性的信息)。

返回值

没有返回值。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

HDM_SETFILTERCHANGETIMEOUT。

HDN_GETDISPINFO

当控件需要获取关于一个信号回叫头标项目的信息时，头标控件的所有者将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
HDN_GETDISPINFO
pDispInfo = (LPNMHDDISPINFO) lParam;
```

参数

pDispInfo：NMHDDISPINFO数据结构的地址。在进行信息输入时，这个数据结构的域将指明需要什么信息以及所感兴趣的项目。

说明

为了返回头标控件信息，必须对这个数据结构的适当数据成员进行填充。如果通告消息将NMHDDISPINFO数据结构的mask数据成员设置为HDI_DI_SETITEM，那么头标控件将存储这些信息，从而在以后不必再次请求这些信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDN_ITEMCHANGED

这个通告消息用来告知头标控件的父窗口，头标项目的属性发生了改变。这个通告消息是以WM_NOTIFY消息形式进行发送的。

```
HDN_ITEMCHANGED
phdr = (LPNMHEADER) lParam;
```

参数

phdr：NMHEADER数据结构的地址，在这个数据结构中包含有关于头标控件信息（这些信息包括已经发生改变的属性信息）。

返回值

没有返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_ITEMCHANGING

这个通告消息用来告知头标控件的父窗口, 头标项目的属性将要改变。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
HDN_ITEMCHANGING  
phdr = (LPNMHEADER) lParam;
```

参数

phdr: NMHEADER数据结构的地址, 在这个数据结构中包含有关于头标控件以及头标项目的信息(这些信息包括将要改变的属性信息)。

返回值

如果允许进行改变, 则返回FALSE; 否则, 返回TRUE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_ITEMCLICK

这个通告消息用来告知头标控件的父窗口, 用户已经单击了本头标控件。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
HDN_ITEMCLICK  
phdr = (LPNMHEADER) lParam;
```

参数

phdr: NMHEADER数据结构的地址, 这个数据结构用来指定头标控件、被单击的头标项目索引以及用来单击这个项目的鼠标按钮。其中, pItem数据成员被设置为NULL。

返回值

没有返回值。

说明

在用户释放鼠标左键之后, 头标控件将发送这个通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_ITEMDBLCLICK

这个通告消息用来告知头标控件的父窗口，用户已经双击了本控件。这个通告消息是以WM_NOTIFY消息的形式进行发送的。只有在头标控件被设置为HDS_BUTTONS样式之后，才发送这个通告消息。

```
HDN_ITEMDBLCLICK
    pnmhdr = (LPNMHEADER) lParam;
```

参数

pnmhdr: NMHEADER数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDN_TRACK

这个通告消息用来告知头标控件的父窗口，用户正在拖动头标控件中的一个分隔标记。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
HDN_TRACK
    phdr = (LPNMHEADER) lParam;
```

参数

phdr: NMHEADER数据结构的地址，在这个数据结构中包含有关于头标控件以及其分隔标记正在被拖动的项目的信息。

返回值

如果需要继续跟踪分隔标记，则返回FALSE；如果需要终止跟踪分隔标记，则返回TRUE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NM_CUSTOMDRAW (头标)

这个通告消息由头标控件向它的父窗口发送，告知父窗口将要进行绘制操作。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CUSTOMDRAW
    lpNMCustomDraw = (LPNMCUSTOMDRAW) lParam;
```

参数

lpNMCustomDraw: NMCUSTOMDRAW数据结构的地址, 在这个数据结构中包含有关于绘制操作的信息。这个数据结构的dwItemSpec数据成员中包含有将要被绘制的项目索引, 而IItemParam数据成员中则包含项目的IParam。

返回值

应用程序能够返回的值取决于当前的绘制阶段。NMCUSTOMDRAW数据结构的dwDrawStage数据成员中保存有用来描述绘制阶段的值。所返回的值必须是下列值之一。

当dwDrawStage等于CDDS_PREPAINT时:

CDRF_DODEFAULT: 表示控件将对自身进行绘制。在这种情况下, 控件不会发送任何其他NM_CUSTOMDRAW消息。

CDRF_NOTIFYITEMDRAW: 表示控件将告知父窗口任何与项目相关的绘制操作。在这种情况下, 控件将在绘制项目之前与之后都发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYITEMERASE: 表示控件将在项目被擦除时告知父窗口。在这种情况下, 控件将在项目的擦除之前与之后发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYPOSTERASE: 表示控件将在某个项目被擦除之后告知父窗口。

CDRF_NOTIFYPOSTPAINT: 表示控件将在绘制某个项目之后告知父窗口。

CDRF_NOTIFYSUBITEMDRAW: 在4.71版本中有效, 它表示控件将在一个列表视图的子项目被绘制时告知父窗口。

CDRF_NEWFONT: 表示应用程序为项目指定了一个新的字体, 并且控件将使用这个新的字体。

CDRF_SKIPDEFAULT: 表示应用程序自行绘制了这个项目, 这时控件将不会绘制这个项目。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.02或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

8.3节。

NM_RCLICK (头标)

这个通告消息用来告知树视图控件的父窗口, 用户已经在控件中单击了鼠标右键。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

NM_RCLICK


```
lpmh = (LPMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

如果不允许使用缺省处理方式, 则返回非零值; 如果允许进行缺省处理, 则返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RELEASEDCAPTURE (头标)

这个通告消息用来告知头标控件的父窗口, 控件正在释放鼠标捕获信息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE
```

```
lpmh = (LPMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

15.4.4 头标控件数据结构

HDHITTESTINFO

在这个数据结构中包含有关于鼠标击中测试的信息, 此数据结构必须与HDM_HITTEST消息一同使用。这个数据结构正在取代HD_HITTESTINFO数据结构。

```
typedef struct _HD_HITTESTINFO {
    POINT pt;
    UINT flags;
    int iItem;
```

```
} HDHITTESTINFO, FAR *LPDHITTESTINFO;
```

成员

pt: POINT数据结构, 在这个数据结构中包含有将要被测试的点(在客户坐标中)。

flags: 这是一个变量, 可以接收关于鼠标击中测试的结果信息。这个数据成员可以是下列一个或多个值:

HHT_ABOVE	表示这个点位于头标控件约束矩形的正上方。
HHT_BELOW	表示这个点位于头标控件约束矩形的正下方。
HHT_NOWHERE	表示这个点在头标控件约束矩形的内部, 但不是头标项目之上。
HHT_ONDIVIDER	表示这个点位于两个头标项目之间的分隔标记上。
HHT_ONDIVOPEN	表示这个点位于宽度为零的项目上。如果拖动这个分隔标记, 将使得项目出现, 而不是将项目的大小调整到分隔标记左端。
HHT_ONHEADER	表示这个点在头标控件的约束矩形中。
HHT_ONFILTER	在5.80版本中有效, 表示这个点正在过滤器区域之上。
HHT_ONFILTERBUTTON	在5.80版本中有效, 表示这个点正在过滤器按钮之上。
HHT_TOLEFT	表示这个点在头标控件约束矩形的左端。
HHT_TORIGHT	表示这个点在头标控件约束矩形的右端。

可以将上述值的二个进行组合, 例如, 当点的位置位于客户区的上方而且在客户区的左端时。

item: 如果鼠标击中测试成功, 那么这个数据成员将包含在集中测试点上的项目的索引。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

HDITEM

这个数据结构中包含有头标控件某个项目的信息, 此数据结构正在取代HD_ITEM数据结构。

```
typedef struct _HDITEM {
    UINT    mask;
    int     cxy;
    LPTSTR  pszText;
    HBITMAP hbm;
    int     cchTextMax;
    int     fmt;
    LPARAM  lParam;
}
#ifdef _WIN32_IE_5_0
#endif
```

```

    int    iImage;
    int    iOrder;
#endif
#if (_WIN32_IE >= 0x0500)
    UINT    type;
    LPVOID  pvFilter;
#endif
)HDITEM, FAR *LPHDITEM;

```

成员

mask: 这个数据成员用来说明数据结构中的其他数据成员中哪些包含有合法数据，或者用来说明哪些数据成员必须被填充。这个数据成员可以是下列值的组合：

标志位	描 述
HDI_BITMAP	表示hbm数据成员合法
HDI_FORMAT	表示fmt数据成员合法
HDI_FILTER	在5.80版本中有效，表示type数据成员与pvFilter数据成员合法。它可以用来过滤掉那些在type数据成员中指定的值
HDI_HEIGHT	表示cxy数据成员合法，并且设定项目的高度

(续)

标 志 位	描 述
HDI_IMAGE	在4.70版本中有效，表示iImage数据成员合法并且设置将要与项目一同显示的图像
HDI_LPARAM	表示lParam数据成员合法
HDI_ORDER	在4.70版本中有效，表示iOrder数据成员合法并且设定项目的顺序值。
HDI_TEXT	表示pszText数据成员与cchTextMax数据成员合法
HDI_WIDTH	表示cxy数据成员合法并且设定项目的宽度

cxy: 项目的宽度或高度。

pszText: 项目字符串的地址。如果这个字符串文本将从控件中获取，那么这个数据成员必须被初始化为指向一个字符缓冲器。如果这个数据成员被设置为LPSTR_TEXTCALLBACK，那么这个控件将发送HDN_GETDISPINFO通告消息，从而请求这个项目的文本信息。

hbm: 项目位图的句柄。

cchTextMax: 项目字符串的长度，单位为字符。如果这个字符串的文本将从控件中获取，那么这个数据成员必须包含由pszText所指定的地址中字符串的字符数。

fmt: 用来指明项目格式的标志位。

可以设置下列标志位，用来指定文本的对齐情况：

标 志 位	描 述
HDF_CENTER	项目的内容居中
HDF_LEFT	项目的内容左对齐
HDF_RIGHT	项目的内容右对齐

可以设置下列标志位，用来控制显示：

标志位	描述
HDF_BITMAP	项目将显示一个位图
HDF_BITMAP_ON_RIGHT	在4.70版本中有效, 位图将显示在文本的右方。这样并不会影响图像列表中的图像被分配给头标项目
HDF_OWNERDRAW	头标控件的所有者将绘制这个项目
HDF_STRING	项目将显示一个字符串

上述值可以与以下值进行组合:

标志位	描述
HDF_IMAGE	在4.70版本中有效, 表示将从一个图像列表中显示图像。可以通过发送HDM_SETIMAGELIST消息方法来指定消息列表, 同时在数据结构中把iImage数据成员设置为图像的索引
HDF_JUSTIFYMASK	将前面表格中列出的三个对齐方式标志位分开
HDF_RTLEADING	在正常情况下, 窗口将按照从左到右的方式显示文本(LTR)。但是, 窗口也可以被修改为像Hebrew或Arabic语言一样按照从右向左(RTL)的方式显示文本。通常, 头标文本是按照与父窗口中相同的方式显示的, 但如果HDF_RTLEADING被设置, 那么头标文本将按照与父窗口相反的方向来显示文本

lParam: 由应用程序所定义的项目数据。

iImage: 在4.70版本中有效, 它是图像列表中一个图像基于0的索引。在头标项目中, 除了显示任何由hbm域所指定的图像之外, 还显示指定的这个图像。如果iImage被设置为I_IMAGECALLBACK, 那么控件将发送HDN_GETDISPINFO通告消息, 以请求这个项目的文本信息。

iOrder: 在4.70版本中有效, 用来表示头标控件中项目从左到右出现的顺序。也就是说, 位于头标控件最左端的项目的顺序值为0, 这个项目右边的项目顺序值为1, 如此等等。

头标控件可以在同一时刻为某个项目显示文本、图像与位图。对齐标志位可用来决定这三者出现的顺序。对于HDF_LEFT或HDF_CENTER, 其顺序是图像、文本与位图。对于HDF_RIGHT, 这个顺序是位图、图像与文本。

type: 在5.80版本中有效, 用来表示由pvFilter所指定的过滤器类型。其可能的类型是:

标志位	描述
HDFT_ISTRING	字符串数据
HDFT_ISNUMBER	数值数据
HDFT_HASNOVALUE	忽略pvFilter值

pvFilter: 在5.80版本中有效, 用来表示由应用程序所定义的数据项目的地址。通过设置type数据成员的标志位值, 可以决定数据过滤器的类型。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

- 头文件：在commctrl.h中进行了声明。

HD_LAYOUT

在这个数据结构中包含有用来设置头标控件大小与位置的信息。HD_LAYOUT数据结构应该与HDM_LAYOUT消息一同使用，这个数据结构用来取代HD_LAYOUT数据结构。

```
typedef struct _HD_LAYOUT {
    RECT FAR* prc;
    WINDOWPOS FAR* pwp;
} HD_LAYOUT, FAR *LPHD_LAYOUT;
```

成员

prc: RECT数据结构的地址，在这个数据结构中包含有头标控件的矩形将要占用的区域坐标。

pwp: WINDOWPOS数据结构的地址，这个数据结构可用来接收关于头标控件正确大小与位置的信息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

HDTEXTFILTER 数据结构

在这个数据结构中包含有关于头标控件文本过滤器的信息。

```
typedef struct _HDTEXTFILTER {
    LPTSTR pszText;
    INT cchTextMax;
} HDTEXTFILTER, FAR *LPHD_TEXTFILTER;
```

成员

pszText: [in], 包含过滤器内容的缓冲器地址。

cchTextMax: [in], 用来说明编辑控件缓冲器最大大小的值，单位为字符。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

NMHHDISPINFO

在这个数据结构中包含有用来处理HDN_GETDISPINFO通告消息的信息。

```
typedef struct tagNMHHDISPINFO {
    NMHDR hdr;
    int iItem;
    UINT mask;
    LPCTSTR pszText;
    int cchTextMax;
    int iImage;
    LPARAM lParam;
} NMHHDISPINFO, FAR* LPNMHHDISPINFO;
```

成员

hdr: NMHDR数据结构, 在这个数据结构中包含有关于本通告消息的信息。

iItem: 头标控件中项目基于0的索引。

mask: 用来说明数据结构中的哪些数据成员将要被头标控件的所有者所填充的标志位集合。

这个值可以是下列值的组合:

HDI_TEXT	表示pszText域必须被填充。
HDI_IMAGE	在4.70版本中有效, 表示iImage域必须被填充。
HDI_LPARAM	表示lParam域必须被填充。
HDI_DI_SETITEM	在4.70版本中有效, 它是一个返回值, 表示头标控件应该存储这个项目的信息, 从而不必进行再次信息请求。

pszText: 一个以null结尾的字符串的地址, 在这个字符串中包含有头标项目将要显示的文本。

cchTextMax: pszText数据成员所指向的缓冲器的大小。

iImage: 图像列表中某个图像基于0的索引。所指定的图像将与头标项目一同显示, 但这个图像并不会取代项目的位图。如果iImage数据成员被设置为I_IMAGECALLBACK, 那么控件将使用HDN_GETDISPINFO通告消息请求关于这个项目的图像信息。

lParam: 一个与项目相关联的32位值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMHHFILTERBTNCLICK 数据结构

这个数据结构用来指定或接收一个过滤器按钮单击动作的属性。

```
typedef struct tagNMHDFILTERBTNCCLICK {
    NMHDR hdr;
    INT item;
    RECT rc;
} NMHDFILTERBTNCCLICK, FAR* LPNMHDFILTERBTNCCLICK;
```

成员

hdr: NMHDR数据结构的句柄, 在这个数据结构中包含有某些附加信息。

item: 这个数据结构所指向的控件基于0的索引。

rc: RECT数据结构的地址, 在这个数据结构中包含有过滤器按钮的客户区矩形。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

HDN_FILTERBTNCCLICK。

NMHEADER

在这个数据结构中包含有关于头标控件通告消息的信息。这个数据结构将用来取代HD_NOTIFY数据结构。

```
typedef struct tagNMHEADER{
    NMHDR hdr;
    int iItem;
    int iButton;
    HDITEM FAR* pItem;
}NMHEADER, FAR* LPNMHEADER;
```

成员

hdr: 包含有关于本通告消息的信息的NMHDR数据结构。

iItem: 作为通告消息焦点的头标项目基于0的索引。

iButton: 用来产生通告消息的鼠标按钮的索引值, 这个数据成员可以是以下的值之一:

- 0 鼠标左按钮
- 1 鼠标右按钮
- 2 鼠标中间按钮

pItem: HDITEM数据结构的可选指针, 在这个数据结构中包含有关于由iItem项目所指定的信息。HDITEM数据结构的mask数据成员用来说明这个数据结构的哪些数据成员是合法的。

说明

虽然大多数头标控件通告消息都将会向NMHEADER数据结构传送一个指针，但只有其中某些通告消息使用pItem数据成员来传送HDITEM数据结构。那些使用pItem数据成员的通告消息可能不会提供关于项目的完全信息。为了获取关于这个项目的更多信息，应该使用HDM_GETITEM。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。



第16章 热键控件

热键控件 (hot-key control) 是一个窗口, 它使得用户能够输入键盘的组合键作为热键。所谓热键, 就是键盘上键的组合, 当用户按下这些组合键时, 就可以快速地执行某个动作。例如, 用户可以创建用来激活一个给定窗口并且将这个窗口放置到屏幕最前端的热键。热键控件可以显示用户的选择, 从而确保用户选择了合法的组合键。

16.1 使用热键控件

当用户输入了一个将要作为热键的组合键时, 那么这些键的名将出现在热键控件中。组合键中的键可以包括修改键 (例如CTRL键、ALT键或SHIFT键) 以及伴随键 (例如字符键、箭头键、功能键等等)。

在用户选择一个组合键之后, 应用程序将从热键控件中获取所输入的组合键, 并且利用这个组合键来在系统中建立一个热键。从热键控件中获取的信息包括用来说明修改键的标志位以及伴随键的虚拟键代码。

应用程序可以使用由热键控件所提供的信息来建立系统全局热键或建立与特定线程相关的热键。系统全局热键与特定的窗口相关联, 它允许用户从系统的任何部分中激活窗口。应用程序可以使用WM_SETHOTKEY消息来设置系统全局热键。无论何时, 只要用户按下了一个系统全局热键, 那么在WM_SETHOTKEY中所指定的窗口将获取一个WM_SYSCOMMAND消息, 这个消息用来指定SC_HOTKEY值。此外, 这个消息还能激活获取本消息的窗口, 直到应用程序调用了WM_SETHOTKEY, 热键将一直保持有效。

与特定线程相关的热键将产生一个被发送给特定线程起始位置的WM_HOTKEY消息, 从而使得这个消息可以在下一次消息循环中被删除。应用程序将使用RegisterHotKey函数来设置与特定线程相关的热键。

16.1.1 创建热键控件

使用CreateWindowEx函数来创建一个热键控件, 在使用这个函数时需要指定HOTKEY_CLASS窗口类。当这个函数向热键控件返回一个句柄时, 那么应用程序通常将可以设置关于非法热键组合的规则, 并且提供一个缺省的组合键。如果应用程序没有设置任何规则, 那么用户就可以选择任何键或者键组合作为热键。但是, 大多数应用程序并不允许用户使用普通键 (例如字符A) 作为热键。

下列函数将创建一个热键控件, 它使用HKM_SETRULES消息与HKM_SETHOTKEY消息来初始化这个热键控件, 然后返回热键控件句柄。这个热键控件不允许用户选择单独的修改键作为热键, 也不允许用户只选择SHIFT键与一个普通键作为热键 (这些规则能够有效地防止用户在输入文本的过程中意外地输入热键)。

```

//InitializeHotkey -creates a hot-key control and sets
// rules and default settings for it.
//Returns the handle of the hot-key control.
//hwndDlg -handle of the parent window (dialog box).
//
//Global variable
// g_hinst -handle of the application instance.
extern HINSTANCE g_hinst;
HWND WINAPI InitializeHotkey(HWND hwndDlg)
{
    //Ensure that the common control DLL is loaded.
    InitCommonControls();

    hwndHot =CreateWindowEx(
        0,                                //no extended styles
        HOTKEY_CLASS,                      //class name
        "",                                //no title (caption)
        WS_CHILD |WS_VISIBLE,             //style
        10,10,                             //position
        200,20,                             //size
        hwndDlg,                           //parent window
        NULL,                               //uses class menu
        g_hinst,                           //instance
        NULL                                //no WM_CREATE parameter
    );

    SetFocus(hwndHot);

    //Set rules for invalid key combinations.If the user
    //does not supply a modifier key,use ALT as a
    //modifier.If the user supplies SHIFT as a modifier
    //key,use SHIFT +ALT instead.
    SendMessage(hwndHot,HKM_SETRULES,
        (WPARAM)HKCOMB_NONE |HKCOMB_S, //invalid key
                                         //combinations
        MAKELPARAM(HOTKEYF_ALT,0));      //add ALT to
                                         //invalid entries
    //Set CTRL +ALT +A as the default hot-key for this window.
    //0x41 is the virtual key code for 'A'.
    SendMessage(hwndHot,HKM_SETHOTKEY,
        MAKEWORD(0x41,HOTKEYF_CONTROL |HOTKEYF_ALT),0);

    return hwndHot;
}

```

16.1.2 热键控件消息

在创建热键控件之后，应用程序将使用三个消息与这个控件进行交互，这三个消息是 HKM_SETRULES消息、HKM_SETHOTKEY消息与HKM_GETHOTKEY消息。

应用程序可以发送HKM_SETRULES消息来指定某些由CTRL键、ALT键与SHIFT键所形成的组合键被视为是非法的热键。如果应用程序指定了一个非法的键组合，那么就应该指定在用户选择非法的键组合时所使用的缺省修改键。如果用户输入了非法键组合，那么系统将对这个非法的键组合与缺省的键组合执行一个逻辑或操作。所得到的结果被视为是合法的键组合，然后被转换为一个字符串并显示在控件中。

HKM_SETHOTKEY消息允许应用程序为热键控件设置热键组合。同时，这个消息通常在热键控件被创建时使用。

应用程序可以使用HKM_GETHOTKEY消息来获取虚拟键代码以及由用户所选择的热键的修改标志。

16.1.3 热键控件通告消息

热键控件并不使用WM_NOTIFY消息来发送任何通告消息。但是，当用户改变控件的内容时，它将通过EN_CHANGE消息来发送WM_COMMAND通告消息。

16.1.4 获取与设置热键控件

在用户改变热键之后，应用程序应该使用HKM_GETHOTKEY消息来从热键控件中获取热键。这个消息将获取一个16位值，在这个值中包含有用来描述热键的虚拟键代码以及修改键。

下列函数将从热键控件中获取键组合，然后使用WM_SETHOTKEY消息来设置系统全局热键。需要说明的是，不能为拥有WS_CHILD窗口样式的窗口设置系统全局热键。

```
//ProcessHotkey -retrieves the hot key from the hot-key
// control and sets it as the hot key for the
// application's main window.
//Returns TRUE if successful,or FALSE otherwise.
//hwndHot -handle of the hot-key control.
//hwndMain -handle of the main window.
BOOL WINAPI ProcessHotkey(HWND hwndHot,HWND hwndMain)
{
    WORD wHotkey;
    UINT iSetResult;

    //Retrieve the hot key (virtual key code and modifiers).
    wHotkey =SendMessage(hwndHot,HKM_GETHOTKEY,0,0);

    //Use the result as wParam for WM_SETHOTKEY.
    iSetResult =SendMessage(hwndMain,WM_SETHOTKEY,wHotkey,0);

    switch (iSetResult)
    {
        case 2: //WM_SETHOTKEY succeeded.
            MessageBox(NULL,"Hot key previously assigned",
                "Okay",MB_OK);
            return TRUE;
    }
}
```

```

case 1:          //WM_SETHOTKEY succeeded.
    return TRUE;

case 0:
    MessageBox(NULL, "Invalid window for hot key",
        "Error", MB_OK);
    return FALSE;

case -1:
    MessageBox(NULL, "Invalid hot key",
        "Error", MB_OK);
    return FALSE;

default:
    MessageBox(NULL, "Unknown error", "Error", MB_OK);
    return FALSE;
}
}

```

16.1.5 缺省热键消息处理过程

本节将讨论窗口过程为热键控件预先定义的HOTKEY_CLASS窗口类所处理的窗口消息。

消 息	执行的动作
WM_CHAR	获取虚拟键代码
WM_CREATE	初始化热键控件, 清除任何热键规则并且使用系统字体
WM_ERASEBKGD	隐藏脱字符 (^), 调用DefWindowProc函数, 然后显示脱字符
WM_GETDLGCODE	返回DLGC_WANTCHARS与DLGC_WANTARROWS值的组合
WM_GETFONT	获取字体
WM_KEYDOWN	如果键是ENTER键、TAB键、SPACE BAR键、DEL键、ESC键或者BACKSPACE键, 那么将调用DefWindowProc函数。如果键是SHIFT键、CTRL键或者ALT键, 那么将检查组合键是否合法, 若是, 将利用组合键设置热键。对于所有其他的键, 将在不进行任何检查的情况下就将它们设置为热键
WM_KEYUP	获取虚拟键代码
WM_KILLFOCUS	析构脱字符
WM_LBUTTONDOWN	将这个窗口设置为焦点
WM_NCCREATE	设置WS_EX_CLIENTEDGE窗口样式
WM_PAINT	绘制热键控件
WM_SETFOCUS	创建与显示脱字符
WM_SETFONT	设置字体
WM_SYSCHAR	获取虚拟键代码
WM_SYSKEYDOWN	如果键是ENTER键、TAB键、SPACE BAR键、DEL键、ESC键或者BACKSPACE键, 那么将调用DefWindowProc函数。如果键是SHIFT键、CTRL键或者ALT键, 那么将检查组合键是否合法, 若是, 将利用组合键设置热键。对于所有其他的键, 将在不进行任何检查的情况下就将它们设置为热键
WM_SYSKEYUP	获取虚拟键代码

16.2 热键控件参考

热键控件消息

HKM_GETHOTKEY

从热键控件中获取某个热键的虚拟键代码与修改键标志。

```
HKM_GETHOTKEY  
    wParam = 0;  
    lParam = 0;
```

返回值

返回虚拟键代码与修改键标志。其中，虚拟键代码位于低位字节，修改键标志位于高位字节。修改键标志可以是下列值的组合：

HOTKEYF_ALT	ALT键
HOTKEYF_CONTROL	CTRL键
HOTKEYF_EXT	扩展键
HOTKEYF_SHIFT	SHIFT键

说明

这个消息所返回的16位值可以用来作为WM_SETHOTKEY消息中的wParam参数。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

HKM_SETHOTKEY

为热键控件设置热键组合键。

```
HKM_SETHOTKEY  
    wParam = MAKEWORD(bVKHotKey, bfMods);  
    lParam = 0;
```

参数

bVKHotKey：热键的虚拟键代码。

bfMods：修改键标志，用来表示与bVKHotKey参数一同组合使用的键，从而定义热键组合键。关于修改键标志值的列表，请参见HKM_GETHOTKEY消息的描述。

返回值

没有返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

WM_GETHOTKEY。

HKM_SETRULES

为热键控件定义非法键组合以及缺省的修改键组合。

HKM_SETRULES

```
wParam = (WPARAM) fwCombInv;
lParam = MAKELPARAM(fwModInv, 0);
```

参数

fwCombInv: 标志的数组, 用来指定非法的键组合。这个参数可以是下列值的组合:

HKCOMB_A	ALT
HKCOMB_C	CTRL
HKCOMB_CA	CTRL+ALT
HKCOMB_NONE	非修改键
HKCOMB_S	SHIFT
HKCOMB_SA	SHIFT+ALT
HKCOMB_SC	SHIFT+CTRL
HKCOMB_SCA	SHIFT+CTRL+ALT

fwModInv: 标志的数组, 用来指定当用户输入一个非法的键组合时所应该使用的键组合。

关于修改键标志值的列表, 请参见HKM_GETHOTKEY消息的描述。

返回值

没有返回值。

说明

当用户输入一个非法的键组合时(由fwCombInv来指定), 系统将使用位或操作来将用户所输入的键与fwModInv中所指定的标志键进行组合, 所得到的结果将被转换为一个字符串, 然后在热键控件中显示。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

第17章 IP地址控件

IP地址控件能够使得用户以一种非常容易理解的格式输入IP地址。这个控件也允许应用程序以数字的形式而不是用文本的形式获取IP地址。

17.1 关于IP地址控件

Microsoft Internet Explorer 4.0版本中引入了IP地址控件，这个控件与编辑控件非常相似，它允许用户按照IP的格式输入数字地址。IP地址格式包括四个由三位数字形成的域，其中的每个域将被分别进行处理，这些域的编号以0开始进行，并且按照从左到右的顺序递增，如图17-1所示。

IP地址控件只允许在其中的每个域中输入数字文本，一旦在给定的域中输入了三个数值，那么键盘焦点将自动转移到下一个域。如果应用程序不需要对整个域进行填充，那么用户就可以输入少于三个的数字。例如，如果域中需要包含21，那么输入21并且按下向右箭头键（RIGHT ARROW）时，将进入下一个域。

每个域的缺省范围是0~255，但应用程序可以将这个范围改变为由（IPM_SETRANGE）消息所指定的范围。

注意 IP地址控件在Comctl32.dll的4.71版本及其后续版本中实现。

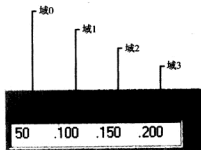


图17-1 IP地址控件

17.2 使用IP地址控件

本节将讨论如何在应用程序中实现IP地址控件。

17.2.1 初始化IP地址控件

如果需要使用IP地址控件，就应该调用InitCommonControlsEx函数，并且在函数调用时设置INITCOMMONCONTROLSEX数据结构dwICC数据成员的ICC_INTERNET_CLASSES标志。

17.2.2 创建IP地址控件

可以使用CreateWindow或CreateWindowEx API函数来创建IP地址控件。这个控件的类名是WC_IPADDRESS，它在Comctl.h头文件中定义。目前还不存在IP地址控件所特有的样式，但是，由于它是一个子控件，因此可以使用WS_CHILD样式对这个控件进行最小化。

17.2.3 IP地址控件不是编辑控件

IP地址控件并不是编辑控件，这个控件将不会对EM_消息进行响应。但是，它将通过WM_COMMAND消息向所有者窗口发送下列编辑控件通告消息。需要说明的是，IP地址控件也可以通过WM_NOTIFY消息来发送专用的IPN_通告消息：

通告消息	产生通告消息的原因
EN_CHANGE	当IP地址控件中的任何域发生改变时，将发送这个通告消息。与标准的编辑控件EN_CHANGE通告消息一样，这个通告消息也是在屏幕被更新之后接收的
EN_KILLFOCUS	当IP地址控件失去键盘焦点时，将发送这个通告消息
EN_SETFOCUS	当IP地址控件获得键盘焦点时，将发送这个通告消息

17.3 IP地址控件参考

17.3.1 IP地址控件消息

IPM_CLEARADDRESS

清除IP地址控件中的内容。

```
IPM_CLEARADDRESS
```

```
wParam = 0;
```

```
lParam = 0;
```

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

IPM_GETADDRESS

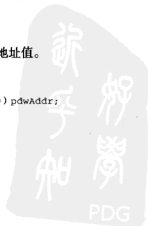
获取IP地址控件中所有四个域的地址值。

```
IPM_GETADDRESS
```

```
wParam = 0;
```

```
lParam = (LPARAM)(LPDWORD) pdwAddr;
```

参数



pdwAddr: 用来接收地址值的DWORD地址值。其中, 3号域的值将包含在DWORD的0到7位, 2号域的值将包含在DWORD的8到15位, 1号域的值将包含在DWORD的16到23位, 0号域的值将包含在DWORD的24到31位。FIRST_IPADDRESS宏、SECOND_IPADDRESS宏、THIRD_IPADDRESS宏与FOURTH_IPADDRESS宏也可用来提取地址信息。对于任何空白域, 将返回0作为地址值。

返回值

返回非空白域的数目。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

IPM_ISBLANK

判断IP地址控件中的所有域是否都为空白。

```
IPM_ISBLANK
    wParam = 0;
    lParam = 0;
```

返回值

如果所有域都为空白, 则返回非零值, 否则返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

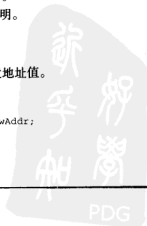
头文件: 在commctrl.h中进行了声明。

IPM_SETADDRESS

为IP地址控件中的所有四个域设置地址值。

```
IPM_SETADDRESS
    wParam = 0;
    lParam = (LPARAM)(DWORD) dwAddr;
```

参数



dwAddr: 包含有新地址值的DWORD值。其中, 3号域的值将包含在DWORD的0到7位, 2号域的值将包含在DWORD的8到15位, 1号域的值将包含在DWORD的16到23位, 0号域的值将包含在DWORD的24到31位。MAKEIPADDRESS宏也可以用来创建地址信息。

返回值

返回值没有被使用。

说明

这个消息并不产生IPN_FIELDCHANGED通告消息。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

IPM_SETFOCUS

为IP地址控件中的指定域设置键盘焦点, 在这个域中的所有文本将被选中。

```
IPM_SETFOCUS
    wParam = (WPARAM) nField;
    lParam = 0;
```

参数

nField: 将要被设置键盘焦点的域的基于0的索引。如果这个值大于域的数目, 那么键盘焦点将被设置给第一个空白域。如果所有的域都不是空白, 那么键盘焦点将被设置给第一个域。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

IPM_SETRANGE

为IP地址控件中指定的域设置合法的范围。

```
IPM_SETRANGE
    wParam = (WPARAM) nField;
    lParam = (LPARAM) (WORD) wRange;
```

参数

nField: 将要被设置范围的域基于0的索引。

wRange: 一个WORD值, 在这个值中的低位字节部分包括设置范围的最低值, 高位字节部分包括设置范围的最高值。这两个值是包含在范围之内的, 也可以使用MAKEIPRANGE宏来创建这个范围值。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

说明

如果用户在域中输入了一个超出所设置范围的值, 控件将发送IPN_FIELDCHANGED通告消息, 在这个通告消息中给出了所输入的值。如果在发送通告消息之后, 这个值仍然超出了范围, 那么控件将把所输入的值修改成为最近的范围限制值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

17.3.2 IP地址控件通告消息

IPN_FIELDCHANGED

当用户改变控件中的某个域或者从一个域转移到另一个域时, 将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
IPN_FIELDCHANGED
    lpnmipa = (LPNMIPADDRESS) lParam;
```

参数

lpnmipa: NMIPADDRESS数据结构的地址, 在这个数据结构中包含有关于已经改变的地址的信息。数据结构的iValue数据成员将包括所输入的值, 即使是这个值超出了设置范围。在响应这个通告消息时, 可以将这个数据成员的值修改为包含在值范围之内的任何值。

返回值

返回值被忽略。

说明

这个通告消息将在响应IPM_SETADDRESS消息的时候进行发送。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

17.3.3 IP地址控件宏

FIRST_IPADDRESS

从一个利用IPM_GETADDRESS消息所获取的IP地址包中提取0号域的值。

```
BYTE FIRST_IPADDRESS (
    LPARAM lParam
);
```

参数

lParam: IP地址包的值。

返回值

返回包含0号域值的BYTE值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

FOURTH_IPADDRESS

从一个利用IPM_GETADDRESS消息所获取的IP地址包中提取3号域的值。

```
BYTE FOURTH_IPADDRESS (
    LPARAM lParam
);
```

参数

lParam: IP地址包的值。

返回值

返回包含3号域值的BYTE值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MAKEIPADDRESS

将四个字节值打包为一个可以由IPM_SETADDRESS消息所使用的单独LPARAM值。

```
LPARAM MAKEIPADDRESS (
    BYTE b0,
    BYTE b1,
    BYTE b2,
    BYTE b3,
);
```

参数

b0: 0号域的地址。

b1: 1号域的地址。

b2: 2号域的地址。

b3: 3号域的地址。

返回值

返回包含地址信息的LPARAM值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MAKEIPRANGE

将两个字节值打包为一个可以由IPM_SETRANGE消息所使用的单独LPARAM值。

```
LPARAM MAKEIPRANGE (
```

```
BYTE low,  
BYTE high  
);
```

参数

low: 范围值的下限。

high: 范围值的上限。

返回值

返回包含范围值的LPARAM值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

SECOND_IPADDRESS

从一个利用IPM_GETADDRESS消息所获取的IP地址包中提取1号域的值。

```
BYTE SECOND_IPADDRESS (  
    LPARAM lParam  
);
```

参数

lParam: IP地址包的值。

返回值

返回包含1号域值的BYTE值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

THIRD_IPADDRESS

从一个利用IPM_GETADDRESS消息所获取的IP地址包中提取2号域的值。

```
BYTE THIRD_IPADDRESS (  

```

```
LPARAM lParam
);
```

参数

lParam: IP地址包的值。

返回值

返回包含2号域值的BYTE值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

17.3.4 IP地址控件数据结构

NMIPADDRESS

在这个数据结构中包含有关于IPN_FIELDCHANGED通告消息的信息。

```
typedef struct tagNMIPADDRESS {
    NMHDR hdr;
    int iField;
    int iValue;
}NMIPADDRESS, *LPNMIPADDRESS;
```

成员

hdr: 包含关于本通告消息更多信息的NMHDR数据结构。

fField: 已经发生改变的域基于0的索引。

iValue: iField数据成员中所指定域的新值。在处理这个IPN_FIELDCHANGED通告消息时, 本数据成员可以被设置为这个域范围中的任何值, 并且控件将把新的值放在这个域中。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

第18章 月历控件

月历控件能够实现一个像日历一样的用户界面，从而为用户进行日期的输入或者选择提供了一种具有引导意义的简易认识方法。这个控件还能够为应用程序提供利用现有的日期类型来获取与设置控件中日期的方法。

18.1 关于月历控件

月历控件在Comctl32.dll的4.70版本及其后续版本中实现。它能够为用户提供一种简单的指导性方法，从一个熟悉的界面中选择日期。图18-1显示了对话框中的一个月历控件。

应用程序通过调用CreateWindowEx函数来创建月历控件，并且在调用这个函数时将把MONTHCAL_CLASS设置为窗口类。当从通用控件动态链接库（DLL）中加载月历类时，将注册这个类。为了注册这个类，必须调用InitCommonControlsEx函数，并且在调用时指定INITCOMMONCONTROLSEX数据结构中的ICC_DATE_CLASSES位标志。

注意 Windows并不支持1601年之前的日期。关于详细信息，请参见FILETIME。

月历控件是基于Gregorian日历产生的，这个日历在1753年被引入。因此，它不能计算在1753年之前就被使用的Julian日历的日期。

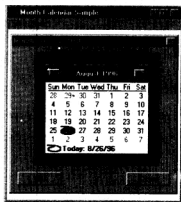


图18-1 月历控件

18.1.1 月历控件用户界面

月历控件的用户界面允许用户从所显示的日历中选择一个日期，或者以各种方法改变控件显示。

1) 滚动进行控件显示。

缺省情况下，当用户单击月历控件顶部左方或者顶部右方的箭头按钮时，控件将更新显示上一个月份或者下一个月份。如果在某个时候，月历控件显示多个月份，那么显示将根据当前正在视图中的月份数来进行改变。也就是说，如果月历控件正在显示1月、2月与3月，这时用户单击了顶部右方的箭头按钮，那么控件将更新显示4月、5月与6月。用户也可以通过单击显示在第一个月份前面的月或者显示在最后一个月份后面的月方法来完成这个动作。

PAGE UP (VK_NEXT)

转移到下一个月份。

PAGE DOWN (VK_PRIOR)

转移到上一个月份。

HOME (VK_HOME)

转移到当前月份的第一天。

END (VK_END)

转移到当前月份的最后一天。

CTRL+HOME

转移到第一个可见的月份。

CTRL+END

转移到最后一个可见的月份。

如果使用MCM_SETMONTHDELTA消息或者相应的MonthCal_SetMonthDelta宏，那么应用程序就可以改变控件将要更新的月份数。但是，PAGE UP键与PAGE DOWN键将只改变所选择月份数到邻接一个月份，而不管所显示的月份数或者由MCM_SETMONTHDELTA所设置的值。

2) 选择一个非顺序的月份。

当用户单击所显示月份的名时，将出现列出一年中所有月份的弹出菜单，用户可以在这个弹出菜单列表中选择一个月份。如果用户的选择不可见，那么月历控件将滚动显示到所选择的月份。

3) 选择一个不同的年份。

如果用户单击了位于月名旁边的年份，那么将在年份的位置上出现一个下拉控件，用户可以利用这个控件来改变年份。当下拉控件失去焦点时，月历控件将更新自己的显示，从而显示所选择的年份。与这个动作相关的键盘命令是：

CTRL+VK_NEXT

转移到下一年。

CTRL+VK_PRIOR

转移到上一年。

4) 选择当前日。

如果月历控件没有使用MCS_NOTODAY样式，那么用户就能够通过单击位于控件底部的“今天”文本来返回到当前日。如果当前日不可见，那么控件将更新显示到当前日。与这个动作相关联的键盘命令是：

VK_LEFT

转移到上一日。

VK_RIGHT

转移到下一日。

VK_UP

转移到上一星期。

VK_DOWN

转移到下一星期。

18.1.2 日状态

使用MCS_DAYSTATE样式的月历控件支持日状态。控件可以使用日状态信息来决定如何在控件中显示某个特定的日。日状态信息可以用一个32位的数据类型MONTHDAYSTATE来表示。MONTHDAYSTATE中的每一位（从1到31）分别表示一个月中每一天的状态。如果相应的位处于打开状态，那么这个日将被显示为黑体；否则，这个日将不被进行突出显示。

应用程序可以通过发送MCM_SETDAYSTATE消息或者通过使用相应的宏MonthCal_SetDayState的方法来特别地设置日状态。此外，使用MCS_DAYSTATE样式的月历控件还可以发送MCN_GETDAYSTATE通告消息用来请求日状态信息。关于支持日状态的更多信息，请参见18.2.2节与18.2.3节。

18.1.3 月历控件样式

月历控件有几种样式（style），可用来决定此控件外观与行为。在利用CreateWindowEx创建月历控件时，应该在dwStyle参数中包括所需的样式信息。

在创建月历控件之后，可以改变除了MCS_DAYSTATE样式与MCS_MULTISELECT样式之外的其他所有样式。为了完成这种样式的改变，必须析构现有的月历控件然后再利用所需的样式创建新的控件。而如果需要获取或者改变任何其他的窗口样式，只要使用GetWinowLong函数与SetWinowLong函数即可。

MCS_MULTISELECT样式的月历控件允许用户选择日范围。缺省情况下，控件将允许用户选择七个连续的日。应用程序可以使用MCM_SETMAXSELCOUNT消息或者相应的MonthCal_SetMaxSelCount宏来改变控件的缺省行为。

当月历控件使用MCS_WEEKNUMBERS样式时，它将在每个月份的左方显示星期号。如果月历控件使用了MCS_NOTODAY样式，那么控件将不再显示用来圈住当前日的圆。

当用户希望控件能够将特定的日期显示为黑体的进行高亮显示时，MCS_DAYSTATE样式就非常有用。关于更多信息，请参见18.1.2节。

18.1.4 本地化

月历控件从LOCALE_USER_DEFAULT中获取月历控件的格式信息以及所有的字符串。对于Windows 2000及其更新系统，月历控件还将从LOCALE_SYEARMONTH获取月份的标题格式。即使是对于相同的DLL版本，根据应用程序所运行的操作系统不同，月历控件的外观也将会有区别。例如，对于Windows NT 4.0，月份的标题被显示为这样的形式：“September 1998”。而在Windows 2000上，月历标题则显示为这样的形式：“September, 1998”。

18.1.5 月历控件通告消息

当月历控件接收到用户的输入或者必须请求日状态信息时（仅仅对于MCS_DAYSTATE样式），那么它将发送通告消息。控件的父窗口将把这些通告消息接收为WM_NOTIFY消息。

月历控件使用下列通告消息：

通告消息	描 述
MCN_GETDAYSTATE	请求将要被显示为黑体的日信息。关于更多信息, 请参见18.2.3 “准备MONTHDAYSTATE数组”
MCN_SELCHANGE	告知父窗口, 所选择的日期或者日期范围发生了改变。当用户显式地改变了当前月中的选择日期或者为了响应前/后月的浏览而隐含地改变了选择日期时, 控件将发送这个通告消息
MCN_SELECT	告知父窗口, 用户明确地选择了某个日期

18.1.6 月历控件中的时间

由于月历控件不能用来选择时间, 因此SYSTEMTIME数据结构中与时间相关的域需要按照不同的方法进行处理。在创建控件之后, 将把当前时间插入到控件的“今天”日期与时间中。

如果程序后来选择了某个时间, 控件要么拷贝时间域, 要么首先检查时间的合法性并且在这个时间不合法时存储当前的缺省时间。下面的列表给出了用来设置日期的消息, 并且说明了消息将如何处理时间域。

通告消息	描 述
MCM_SETCURSEL	控件将拷贝时间域, 而不进行任何合法性检查或者做任何修改
MCM_SETRANGE	被传递数据结构中的时间域将被进行合法性检查。如果时间域合法, 那么将不进行任何修改地拷贝。如果时间域不合法, 那么控件将从“今天”日期与时间中拷贝时间域
MCM_SETSELRANGE	被传递数据结构中的时间域将被进行合法性检查。如果时间域合法, 那么将不进行任何修改地拷贝。如果时间域不合法, 那么控件将维持从当前选择范围中的时间域
MCM_SETTODAY	控件将拷贝时间域, 而不进行任何合法性检查或者作任何修改

当从月历控件中获取了一个日期之后, 时间域从所存储的时间中不进行任何修改地拷贝。控件所提供的对月历控件处理的支持, 主要是为了向程序员提供设计方便性。对于除了上面所列出的操作之外的操作结果, 月历控件将不对时间域进行任何检查或修改。

18.2 使用月历控件

本节将给出实现月历控件的信息以及示例代码。

18.2.1 创建月历控件

如果需要创建一个月历控件, 就应该使用CreateWindowEx函数, 并且将MONTHCAL_CLASS指定为窗口类。为此, 首先必须调用InitCommonControlsEx函数, 并且在相应的INITCOMMONCONTROLSEX数据结构中指定ICC_DATE_CLASSES位的值, 从而对窗口类进行注册。

下面的例子演示了如何在一个现有的非模态对话框中创建月历控件。需要说明的是, 传递给CreateWindowEx的大小值全部为0, 这是因为最小的所需大小取决于控件所使用的字体, 而DoNotify例子函数中使用MonthCal_GetMinReqRect宏来请求大小信息, 然后再调用SetWindowPos来改变控件的大小。如果用WM_SETFONT来改变字体, 那么控件的大小将不会作

任何修改。必须再次调用MonthCal_GetMinReqRect函数，并且调整控件大小以适合新的字体：

```
// CreateMonthCal --Creates a month-calendar control in a dialog
// box.Returns the handle to the month-calendar
// control if successful,or NULL otherwise.
//
// hwndOwner --Handle to the owner of the dialog box.
// g_hinst --Global handle to the program instance.
//
/////
HWND WINAPI CreateMonthCal(HWND hwndOwner)
{
    HWND hwnd;
    RECT rc;
    INITCOMMONCONTROLSEX icex;
    //Load the window class.
    icex.dwSize =sizeof(icex);
    icex.dwICC =ICC_DATE_CLASSES;
    InitCommonControlsEx(&icex);

    //Create a modeless dialog box to hold the control.
    g_hwndDlg =CreateDialog(g_hinst,
        MAKEINTRESOURCE(IDD_DIALOG1),
        hwndOwner,
        DlgProc);

    //Create the month-calendar.
    hwnd =CreateWindowEx(0,
        MONTHCAL_CLASS,
        "",
        WS_BORDER |WS_CHILD |WS_VISIBLE |MCS_DAYSTATE,
        0,0,0,0, //resize it later
        g_hwndDlg,
        NULL,
        g_hinst,
        NULL);

    //Get the size required to show an entire month.
    MonthCal_GetMinReqRect(hwnd,&rc);

    //Arbitrary values
    #define LEFT 35
    #define TOP 40

    //Resize the control now that the size values have
    //been obtained.
    SetWindowPos(hwnd,NULL,TOP,LEFT,
        LEFT +rc.right,TOP +rc.bottom,
```

```

SWP_NOZORDER);

//Set colors for aesthetics.
MonthCal_SetColor(hwnd,MCSC_BACKGROUND,RGB(175,175,175));
MonthCal_SetColor(hwnd,MCSC_MONTHBK,RGB(248,245,225));

return(hwnd);
}

```

18.2.2 处理MCN_GETDAYSTATE通告消息

为了请求关于可见月份中日如何显示的信息，月历控件将发送MCN_GETDAYSTATE通告消息。下列由应用程序所定义的DoNotify函数将处理MCN_GETDAYSTATE消息，它将MONTHDAYSTATE数组的值设置为在每个月的第15天为高亮状态。

DoNotify将从lParam所指向的NMDAYSTATE数据结构的cDayState数据成员中提取MONTHDAYSTATE值。然后，将使用由应用程序所定义的BOLDDAY宏来对数组中每个元素的第15位进行循环设置。

```

BOOL WINAPI DoNotify(HWND hwnd,UINT msg,WPARAM wParam,LPARAM lParam)
{
#define BOLDDAY(ds,iDay)if(iDay>0 &&iDay<32)\
    (ds)!=(0x00000001<<(iDay-1))

#define lpmDS ((NMDAYSTATE *)lParam)
#define MAX_MONTHS 12

    MONTHDAYSTATE mds [MAX_MONTHS ];
    INT i,iMax;
    LPNMHDR hdr =(LPNMHDR)lParam;

    switch(hdr->code){
        case MCN_GETDAYSTATE:
            iMax=lpmDS->cDayState;

            for(i=0;i<iMax;i++){
                mds [i ] =(MONTHDAYSTATE)0;
                BOLDDAY(mds [i],15);
            }
            lpmDS->prgDayState =mds;
            break;
    }
    return FALSE;
}

```

18.2.3 准备MONTHDAYSTATE数组

MCM_SETDAYSTATE消息与MCN_GETDAYSTATE通告消息为了决定日期应该如何被显

示，都需要使用MONTHDAYSTATE值数组。控件所显示的每个月份都必须在数组中有相应的元素对应。

为了支持这些消息，应用程序必须能够正确地准备这个数组。下面给出了一个简单的宏，用来为每个月中的给定日所对应的位设置MONTHDAYSTATE值。

```
#define BOLDDAY(ds, iDay) if (iDay > 0 && iDay < 32) \
    (ds) |= (0x00000001 << (iDay-1))
```

利用这个宏，应用程序可以非常简单地对一个由重要的日期组成的数组进行循环，从而在相应的数组元素中设置位。当然，这种处理方法并不是最有效的，但它确实能够完成许多功能。只要应用程序能够正确地设置MONTHDAYSTATE，那么关于这些位的设置方式并不重要。

18.3 月历控件样式

下面给出的是月历控件中所使用的样式：

1. MCS_DAYSTATE

在4.70版本中有效，月历控件将发送MCN_GETDAYSTATE通告消息来请求关于将要哪天显示为黑体的信息。关于支持这个样式的更多信息，请参见18.2.2“处理MCN_GETDAYSTATE通告消息”。

2. MCS_MULTISELECT

在4.70版本中有效，月历控件将允许用户在这个控件中选择一个日期范围。缺省情况下，最大的日期范围是一个星期。可以使用MCM_SETMAXSELCOUNT消息来改变被选择的最大日期范围。

3. MCS_NOTODAY

在4.70版本中有效，月历控件将不显示出现在控件底部的“今天”日期。

4. MCS_NOTODAYCIRCLE

在4.70版本中有效，月历控件将不把“今天”日期用圆圈圈起来。

5. MCS_WEEKNUMBERS

在4.70版本中有效，月历控件将在天的每个行的左边显示星期编号（1-52）。其中，第一周被定义为第一个包含至少四天的那个星期。

18.4 月历控件中的日数字

下面的列表中包含有一些数字，用来表示在月历控件中与星期相对应：

数字值	星期中的天
0	周一
1	周二
2	周三
3	周四
4	周五
5	周六
6	周日

18.5 月历控件参考

18.5.1 月历控件消息

MCM_GETCOLOR

为月历控件中的给定部分获取颜色。可以显式地发送这个消息，也可以使用MonthCal_GetColor宏来发送这个消息。

```
MCM_GETCOLOR
wParam = (WPARAM)(INT) iColor;
lParam = 0;
```

参数

iColor: 用来指定将要获取哪个月历控件颜色的INT值，这个值可以是以下之一：

MCSC_BACKGROUND	获取显示在月份之间的背景色。
MCSC_MONTHBK	获取显示在月份内部的背景色。
MCSC_TEXT	获取用来在月份中显示文本的颜色。
MCSC_TITLEBK	获取在月历标题中显示的背景色。
MCSC_TITLETEXT	获取用来在月历标题中显示文本的颜色。
MCSC_TRAILINGTEXT	获取用来显示第一日与最后一日文本的颜色。第一日与最后一日是指月历控件中上一个月份的最后一日与下一个月份的开始一日。

返回值

如果操作成功，则返回COLORREF值，用来表示为月历控件中的指定部分设置的顏色。否则，将返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_GETCURSEL

获取当前选中的日期。可以显式地发送这个消息，也可以使用MonthCal_GetCurSel宏来发送这个消息。

```
MCM_GETCURSEL
wParam = 0;
```

```
lParam = (LPARAM)(LPSYSTEMTIME) lpSysTime;
```

参数

lpSysTime: SYSTEMTIME数据结构的地址, 这个数据结构可以用来获取当前选中的日期信息。这个参数必须是一个合法的地址, 而不能为NULL。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。如果将这个消息应用于MCS_MULTISELECT样式的月历控件, 那么消息将总会失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_GETFIRSTDAYOFWEEK

为月历控件获取星期的第一天。可以特别地发送这个消息, 也可以使用MonthCal_GetFirstDayOfWeek宏来发送这个消息。

```
MCM_GETFIRSTDAYOFWEEK
wParam = 0;
lParam = 0;
```

返回值

返回包含两个值的DWORD值。如果星期中的第一天被设置为不是LOCALE_IFIRSTDAYOFWEEK的值, 那么这个DWORD值的高位部分将是一个不为0的BOOL值; 否则将返回0。DWORD值的低位部分是一个INT值, 用来表示星期中的第一天, 它将是day numbers中的一个值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_GETMAXSELCOUNT

获取可以在月历控件中选择的最大日期范围。可以显式地发送这个消息，也可以使用MonthCal_GetMaxCount宏来发送这个消息。

```
MCM_GETMAXSELCOUNT
    wParam = 0;
    lParam = 0;
```

返回值

返回一个INT值，用来表示可以为控件所选中的最大总天数。

说明

可以利用MCM_SETMAXSELCOUNT消息来改变可以被选中的最大日期范围。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_GETMAXTODAYWIDTH

获取月历控件中“今天”字符串的最大宽度, 这个宽度应该包含标号文本与日期文本。可以显式地发送这个消息, 也可以使用MonthCal_GetMaxTodayWidth宏来发送这个消息。

```
MCM_GETMAXTODAYWIDTH
    wParam = 0;
    lParam = 0;
```

返回值

返回“今天”字符串的宽度, 单位为像素。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_GETMINREQRECT

获取用来在月历控件中显示完整的一个月份所需的最小尺寸。可以特别地发送这个消息,

也可以使用MonthCal_GetMinReqRect宏来发送这个消息。

```
MCM_GETMINREQRECT
wParam = 0;
lParam = (LPARAM)(LPRECT) lpRectInfo;
```

参数

lpRectInfo: RECT数据结构的地址, 在这个数据结构中可以用来包含约束矩形的信息, 这个参数必须是一个合法的地址, 而不能是NULL。

返回值

如果操作成功, 则返回非零值, 并且lpRectInfo将接收到可应用的约束信息。否则, 这个消息将返回0。

说明

月历控件用来显示一个完整的月份所需的最小窗口大小取决于当前选择的字体、控件样式、系统度量、区域设置。应用程序对上述任何内容的改变都将会影响到最小窗口尺寸, 或者, 在处理WM_SETTINGCHANGE消息时, 应该发送MCM_GETMINREQRECT来判断新的最小尺寸。

注意 由MCM_GETMINREQRECT所返回的矩形并不包括“今天”字符串的宽度。如果没有设置MCS_NOTODAY样式, 应用程序将通过发送MCM_GETMAXTODAYWIDTH消息来获取定义了“今天”字符串宽度的矩形。为了保证“今天”字符串没有被截取, 应该使用这两个矩形中较大的那一个。

由lpRectInfo所指向的数据结构的top数据成员与left数据成员总是为0。而right数据成员与bottom数据成员分别用来代表控件所需的最小cx与cy。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_GETMONTHDELTA

获取月历控件的滚动频率, 作为滚动频率, 是指当用户单击滚动按钮时, 控件移动自己显示月份的数目。可以显式地发送这个消息, 也可以使用MonthCal_GetMonthDelta宏来发送这个消息。

```
MCM_GETMONTHDELTA
wParam = 0;
lParam = 0;
```

返回值

如果先前已经使用了MCM_SETMONTHDELTA消息来设置月份的增量信息,那么将返回用来代表月历当前滚动频率的INT值。如果先前没有使用MCM_SETMONTHDELTA消息设置月份增量,或者月份增量被重新设置为缺省值,那么将返回当前可见的月份数INT值。

环境

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_GETMONTHRANGE

使用SYSTEMTIME数据结构获取用来表示月历控件应该显示的最大与最小范围限制数据信息。可以显式地发送这个消息,也可以使用MonthCal_GetMonthRange宏来发送这个消息。

MCM_GETMONTHRANGE

wParam = (WPARAM)(DWORD) dwFlag;

lParam = (LPARAM)(LPSYSTEMTIME) lprgSysTimeArray;

参数

dwFlag: 指定将要获取的范围限制的值,这个值必须是以下之一:

GMR_DAYSTATE 在这个值中包含有只是部分被显示出来的可视的第一个月与结尾一个月。

GMR_VISIBLE 在这个值中包含有那些完全被显示出来的月份。

lprgSysTimeArray: SYSTEMTIME数据结构数组(二个元素)的地址,这个数组可用来获取dwFlag所指定范围的最低限制值与最高限制值。其中,最低限制值与最高限制值分别放置在lprgSysTimeArray[0]与lprgSysTimeArray[1]中。这个参数必须是一个合法的地址,并且不能为NULL。

返回值

返回由lprgSysTimeArray数组所指定的两个限制之间月份范围的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_GETRANGE

获取为月历所设置的最小允许日期与最大允许日期。可以显式地发送这个消息，也可以使用MonthCal_GetRange宏来发送这个消息。

```
MCM_GETRANGE
wParam = 0;
lParam = (LPARAM)(LPSYSTEMTIME) lprgSysTimeArray;
```

参数

lprgSysTimeArray: SYSTEMTIME数据结构数组（两个元素）的地址，这个数组可用来接收日期限制信息。其中，最小日期限制设置在lprgSysTimeArray[0]中，最大日期限制设置在lprgSysTimeArray[1]中。如果这两个元素中的有一个被设置为全0，那么就不会为月历控件设置相应的限制。这个参数必须是一个合法的地址，并且不能为NULL。

返回值

返回一个DWORD值，这个值可以是0（表示没有任何限制设置）或者为以下用来指定限制信息的值：

GDTR_MAX	为月历控件所设置的最大限制值：这时，lprgSysTimeArray[0]合法并且包含有可应用的日期信息。
GDTR_MIN	为月历控件所设置的最小限制值：这时，lprgSysTimeArray[1]合法并且包含有可应用的日期信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_GETSELRange

获取当前由用户所选中的日期范围上限与下限数据信息。可以显式地发送这个消息，也可以使用MonthCal_GetSelRange宏来发送这个消息。

```
MCM_GETSELRange
wParam = 0;
lParam = (LPARAM)(LPSYSTEMTIME) lprgSysTimeArray;
```

参数

lprgSysTimeArray: SYSTEMTIME数据结构数组(包含两个元素)的地址,这个数组可用来接收用户所选择的最大限制与最小限制。最小限制与最大限制分别放置在lprgSysTimeArray[0]与lprgSysTimeArray[1]中。这个参数必须是一个合法的地址,并且不能为NULL。

返回值

如果操作成功,则返回非零值,否则返回0。如果应用于没有使用MCS_MULTISELECT样式的月历控件MCM_GETSELRANGE将失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_GETTODAY

获取月历控件中被指定为“今天”这个日期的数据信息。可以显式地发送这个消息,也可以使用MonthCal_GetToday宏来发送这个消息。

```
MCM_GETTODAY
    wParam = 0;
    lParam = (LPARAM)(LPSYSTEMTIME) lpToday;
```

参数

lpToday: SYSTEMTIME数据结构的地址,在这个数据结构中可以接收所需的数据信息。这个参数必须是一个合法的地址,并且不能为NULL。

返回值

若操作成功,则返回一个非零值;否则,返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_GETUNICODEFORMAT

获取月历控件的UNICODE字符格式标志。可以显式地发送这个消息，也可以使用MonthCal_GetUnicodeFormat宏来发送这个消息。

```
MCM_GETUNICODEFORMAT  
wParam = 0;  
lParam = 0;
```

返回值

返回月历控件的UNICODE格式标志。如果这个值为非零，那么表示控件正在使用UNICODE字符；如果这个值为0，那么表示控件正在使用ANSI字符。

说明

关于这个消息的更多讨论，请参见CCM_GETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

MCM_SETUNICODEFORMAT。

MCM_HITTEST

判断月历控件的哪个部分正处于屏幕的给定点上。可以显式地发送这个消息，也可以使用MonthCal_HitTest宏来发送这个消息。

```
MCM_HITTEST  
wParam = 0;  
lParam = (LPARAM)(PMCHITTESTINFO) pMCHitTest;
```

参数

pMCHitTest：MCHITTESTINFO数据结构的地址。当发送这个消息时，cbSize数据成员必须被设置为MCHITTESTINFO数据结构的大小，而pt数据成员必须被设置为将进行鼠标集中测试的点。

返回值

在由pMCHitTest所指向的MCHITTESTINFO数据结构成员中设置值，并且返回包含一个或多个下列值的DWORD值：

MCHT_CALENDAR	表示给定的点在月历中。
MCHT_CALENDARBK	表示给定的点在月历的背景中。
MCHT_CALENDARDATE	表示给定的点在月历的某个特定日期上。 pMCHitTest->st上的SYSTEMTIME数据结构将被设置为给定点上的这个日期。
MCHT_CALENDARDATENEXT	表示给定的点在下一个月份的某个日期上(这个月份被部分地显示在当前显示月份的末端)。如果用户单击这个位置,那么月历控件将滚动显示到下一个月份或者月份集合中。
MCHT_CALENDARDATEPREV	表示给定的点在上一个月份的某个日期上(这个月份被部分地显示在当前显示月份的末端)。如果用户单击这个位置,那么月历控件将滚动显示到上一个月份或者月份集合中。
MCHT_CALENDARDAY	表示给定的点在日期缩写上(例如Fri)。这时,pMCHitTest->st上的SYSTEMTIME数据结构将被设置为顶部行的相应日期。
MCHT_CALENDARWEEKNUM	表示给定的点在星期编号上(仅仅对MCS_WEEKNUMBERS样式有效)。这时,pMCHitTest->st上的SYSTEMTIME数据结构将被设置为最左端列上的相应日期。
MCHT_NEXT	表示给定的点在将要导致月历滚动显示到下一个月份或者月份集合上的区域中。这个标志值可用来修改其他鼠标击中测试标志。
MCHT_NOWHERE	表示给定的点不在月历控件上,或者表示给定的点在月历控件的非活跃部分。
MCHT_PREV	表示给定的点在将要导致月历控件滚动到上一个月份或月份集合的区域上。这个标志位值可用来修改其他鼠标器中测试值。
MCHT_TITLE	表示给定的点在月份的标题上。
MCHT_TITLEBK	表示给定的点在月份标题的背景上。
MCHT_TITLEBTNNEXT	表示给定的点在控件右上角的按钮上。如果用户单击了这个点,那么月历控件将滚动显示到下一个月份或者月份集合中。
MCHT_TITLEBTNPREV	表示给定的点在控件左上角的按钮上。如果用户单击了这个点,那么月历控件将滚动显示到上一个月份或者月份集合中。
MCHT_TITLEMONT	表示给定的点在月份标题条的一个月份名上。

MCHT_TITLEYEAR 表示给定的点在月份标题条的一个年份值上。
MCHT_TODAYLINK 表示给定的点在月历控件底部的“今天”链接上。
 pMCHitTest上MCHITTESTINFO数据结构的uHit数据成员将等于返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_SETCOLOR

为月历控件中的给定部分设置颜色。可以显式地发送这个消息, 也可以使用MonthCal_SetColor宏来发送这个消息。

```
MCM_SETCOLOR
    wParam = (WPARAM)(INT) iColor;
    lParam = (LPARAM)(COLORREF) clr;
```

参数

iColor: 用来指定将要设置的月历颜色的INT值, 这个值必须是以下之一:

MCSC_BACKGROUND 这个值将用来设置显示在月份之间的背景色。
MCSC_MONTHBK 这个值将用来设置显示在月份内部的背景色。
MCSC_TEXT 这个值将用来设置在月份内部显示文本的颜色。
MCSC_TITLEBK 这个值可用来设置显示在月历标题中的背景色。
MCSC_TITLETEXT 这个值可用来设置显示在月历标题中的文本颜色。
MCSC_TRAILINGTEXT 这个值可用来设置显示第一日与最后一日的文本颜色。在这里, 第一日与最后一日是指当前月历中显示从上一个月份到下一个月份的日期。

clr: COLORREF的值, 这个值表示将要为月历控件中的指定区域设置的颜色。

返回值

如果操作成功, 则返回COLORREF值, 这个值表示月历控件中指定区域的先前颜色。否则, 返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的

Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_SETCURSEL

设置为月历控件所选择的当前日期。如果所指定的日期不在视图范围内, 控件将更新显示以便把所指定的日期显示出来。可以显式地发送这个消息, 也可以使用MonthCal_SetCurSel宏来发送这个消息。

```
MCM_SETCURSEL
    wParam = 0;
    lParam = (LPARAM)(LPSYSTEMTIME) lpSysTime;
```

参数

lpSysTime: SYSTEMTIME数据结构的地址, 在这个数据结构中包含有将要被设置作为当前选择的日期。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。如果这个消息被应用于设置为MCS_MULTISELECT样式的月历控件, 将产生失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_SETDAYSTATE

为月历控件中当前可见的所有月份设置日状态。可以显式地发送这个消息, 也可以使用MonthCal_SetDayState宏来发送这个消息。

```
MCM_SETDAYSTATE
    wParam = (WPARAM) iMonths;
    lParam = (LPARAM)(LPMONTHDAYSTATE) lpDayStateArray;
```

参数

iMonths: 用来说明在lpDayStateArray所指向的数组中有多少个元素的值。

lpDayStateArray: MONTHDAYSTATE值数组的地址, 这个数组值可用来定义月历控件应该

在自己的显示中如何绘制每个日。

返回值

若操作成功，则返回一个非零值；否则，返回0。

说明

lpDayStateArray数组中必须包含由下列宏所返回的所有值：

```
MonthCal_GetMonthRange( hwndMC, GMR_DAYSTATE, NULL );
```

需要注意的是，lpDayStateArray上的数组中必须包含与月历控件中当前所显示的所有月份相对应的MONTHDAYSTATE值（按照年历顺序排列）。在这种情况下，应该包含被部分显示出来的那两个月份。关于准备数组的更多信息，请参见18.2.3“准备MONTHDAYSTATE数组”。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

MCM_SETFIRSTDAYOFWEEK

为月历控件设置星期中的第一天。可以显式地发送这个消息，也可以使用MonthCal_SetFirstDayWeek宏来发送这个消息。

```
MCM_SETFIRSTDAYOFWEEK
```

```
wParam = 0;
```

```
lParam = (LPARAM)(INT) iDay;
```

参数

iDay：一个INT值，用来表示哪天将被设置为星期中的第一日，这个值必须是一个日编号。

返回值

返回一个包含有两个值的DWORD值。其中，如果先前设置的星期中第一日不等于LOCALE_IFIRSTDAYOFWEEK，那么这个DWORD值的高位字部分将是一个非零的BOOL值，否则为0。DWORD值的低位字部分是一个INT值，用来表示先前所设置的星期中第一日。

说明

如果星期中的第一日被设置为一个不是缺省值（LOCALE_IFIRSTDAYOFWEEK）的值，那么控件将根据所做的改变自动地更新星期中的第一日。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_SETMAXSELCOUNT

设置可以在月历控件中选择的最大天数。可以显式地发送这个消息, 也可以使用MonthCal_SetMaxSelCount宏来发送这个消息。

```
MCM_SETMAXSELCOUNT
    wParam = (WPARAM)(INT) iMax;
    lParam = 0;
```

参数

iMax: 将要被设置为可选择最大天数的INT值。

返回值

如果操作成功, 则返回非零值, 否则, 返回0。如果这个消息应用于被设置为没有使用MCS_MULTISELECT样式的月历控件, 将产生失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_SETMONTHDELTA

为月历控件设置滚动频率。所谓滚动频率是指当用户单击滚动按钮时控件将要转移进行显示的月份数。可以显式地发送这个消息, 也可以使用MonthCal_SetMonthDelta宏来发送这个消息。

```
MCM_SETMONTHDELTA
    wParam = (WPARAM)(INT) iDelta;
    lParam = 0;
```

参数

iDelta: 这是一个表示将要被设置为控件滚动频率的月份数。如果这个值为0, 那么月份增量将被设置为缺省值 (在控件中显示的月份数目)。

返回值

返回表示先前滚动频率的INT值。如果先前没有设置滚动频率, 那么将返回0。

说明

PAGE UP键与PAGE DOWN键, 以及VK_PRIOR与VK_NEXT按钮都可以用来改变所选择的月份一次, 而不管所选择的月份数或由MCM_SETMONTHDELTA所设置的值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCM_SETRANGE

设置月历控件的最小允许日期与最大允许日期。可以显式地发送这个消息, 也可以使用MonthCal_SetRange宏来发送这个消息。

```
MCM_SETRANGE
    wParam = (WPARAM)(SHORT) fWhichLimit;
    lParam = (LPARAM)(LPSYSTEMTIME) lprgSysTimeArray;
```

参数

fWhichLimit: 用来说明哪个日期限制将被设置的标志值, 这个值必须是下列一个或两个值:

GDTR_MAX 将要被设置的最大允许日期。lprgSysTimeArray[1]上的SYSTEMTIME数据结构必须包含有日期信息。

GDTR_MIN 将要被设置的最小允许日期。lprgSysTimeArray[0]上的SYSTEMTIME数据结构必须包含有日期信息。

lprgSysTimeArray: SYSTEMTIME数据结构数组 (两个元素) 的地址, 在这个数组中包含有日期限制信息。如果指定了GDTR_MAX, 那么必须在lprgSysTimeArray[1]中存在最大日期限制设置; 如果指定了GDTR_MIN, 那么必须在lprgSysTimeArray[0]中存在的最小日期限制设置。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_SETSELRange

为月历控件设置所做的选择给一个指定的日期范围。可以显式地发送这个消息，也可以使用MonthCal_SetSelRange宏来发送这个消息。

```
MCM_SETSELRange
wParam = 0;
lParam = (LPARAM)(LPSYSTEMTIME) lpSysTimeArray;
```

参数

lpSysTimeArray: SYSTEMTIME数据结构数组（两个元素）的地址，在这个数组中包含有表示选择限制的日期信息。第一个所选择的日期必须在prgSysTimeArray[0]中指定，最后一个被选择的日期必须在prgSysTimeArray[1]中指定。

返回值

若操作成功，则返回一个非零值；否则，返回0。如果应用于没有使用MCS_MULTISELECT样式的月历控件，那么这个消息将失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_SETTODAY

为月历控件设置“今天”选择。可以显式地发送这个消息，也可以使用MonthCal_SetToday宏来发送这个消息。

```
MCM_SETTODAY
wParam = 0;
lParam = (LPARAM)(LPSYSTEMTIME) lpSysTime;
```

参数

lpSysTime: SYSTEMTIME数据结构的地址，在这个数据结构中包含有将要为控件设置为“今天”选择的数据。如果这个参数被设置为NULL，那么控件将返回缺省设置。

返回值

这个消息的返回值没有使用。

说明

如果“今天”选择没有被设置为缺省设置之外的任何日期，那么以下情况将适用：

- 当时间经过了当天的午夜时，控件将不会自动地更新“今天”选择。
- 控件将不会自动地根据本地设置改变而更新。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

18.1.6节。

MCM_SETUNICODEFORMAT

为控件设置UNICODE字符格式标志。这个消息允许在运行时改变由控件所使用的字符集，而不必重新创建控件。可以显式地发送这个消息，也可以使用MonthCal_SetUnicodeFormat宏来发送这个消息。

```
MCM_SETUNICODEFORMAT
    wParam = (WPARAM)(BOOL) fUnicode;
    lParam = 0;
```

参数

fUnicode：用来决定被控件所使用的字符集。如果这个值非零，那么控件将使用UNICODE字符。如果这个值为0，那么控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式标志。

说明

关于这个消息的更多讨论，请参见CCM_SETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

MCM_GETUNICODEFORMAT。

18.5.2 月历控件宏

MonthCal_GetColor

为月历控件给定部分获取颜色。可以使用这个宏，也可以通过显式发送MCM_GETCOLOR消息的方法来完成此动作。

```
COLORREF MonthCal_GetColor (
    HWND hwndMC,
    INT iColor
);
```

参数

hwndMC: 月历控件的句柄。

iColor: 用来指定将要获取哪个月历控件颜色的INT值，这个值可以是以下之一：

MCSC_BACKGROUND	获取显示在月份之间的背景色。
MCSC_MONTHBK	获取显示在月份内部的背景色。
MCSC_TEXT	获取月份中所显示的文本颜色。
MCSC_TITLEBK	获取显示在月历标题中的背景色。
MCSC_TITLETEXT	获取月历标题中显示的文本颜色。
MCSC_TRAILINGTEXT	获取用来显示第一日与最后一日文本的颜色。第一日与最后一日分别是指当前月历控件中所出现的上一个月份的最后一日与下一个月份的第一日。

返回值

如果操作成功，则返回COLORREF值，用来表示月历控件中指定部分的颜色设置。否则，返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetCurSel

获取当前所选择的日期。可以使用这个宏，也可以通过显式发送MCM_GETCURSEL消息的方法来完成此动作。

```

BOOL MonthCal_GetCurSel (
    HWND hwndMC,
    LPSYSTEMTIME lpSysTime
);

```

参数

hwndMC: 月历控件的句柄。

lpSysTime: SYSTEMTIME数据结构的地址, 在这个数据结构中可以获取当前选择的日期信息。这个参数必须是一个合法的地址, 并且不能为NULL。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。当这个宏应用于被设置为MCS_MULTISELECT样式的月历控件时, 宏总是会失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetFirstDayOfWeek

获取一个月历控件中星期的第一天。可以使用这个宏, 也可以通过显式发送MCM_GETFIRSTDAYOFWEEK消息的方法来完成此动作。

```

DWORD MonthCal_GetFirstDayOfWeek (
    HWND hwndMC
);

```

参数

hwndMC: 月历控件的句柄。

返回值

返回一个包含有两个值的DWORD值。如果信息中的第一天被设置为不是LOCALE_IFIRSTDAYOFWEEK的其他值, 那么这个DWORD值的高位字部分将是一个非零的BOOL值, 否则为0。这个DWORD值的低位字部分是一个表示星期中第一天的INT值, 它将是天的编号。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetMaxSelCount

获取月历控件中可以被选择的最大日期范围。可以使用这个宏, 也可以通过显式发送MCM_GETMAXSELCOUNT消息的方法来完成此动作。

```
DWORD MonthCal_GetMaxSelCount (  
    HWND hwndMC  
);
```

参数

hwndMC: 月历控件的句柄。

返回值

返回能够为月历控件所选择的最大总天数INT值。

说明

可以利用MCM_SETMAXSELCOUNT消息来改变可以被选择的最大日期范围。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetMaxTodayWidth

获取月历控件中“今天”字符串的最大宽度, 这个宽度包含标号文本与日期文本在内。可以使用这个宏, 也可以通过显式发送MCM_GETMAXTODAYWIDTH消息的方法来完成此动作。

```
DWORD MonthCal_GetMaxTodayWidth (  
    HWND hwndMC  
);
```

参数

hwndMC: 月历控件的句柄。

返回值

返回“今天”字符串的宽度, 单位为像素。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版

本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetMinReqRect

获取在月历控件中显示一个完整的月份所需要的最小尺寸大小, 这个大小信息是以RECT数据结构的形式来表示的。可以使用这个宏, 也可以通过显式发送MCM_GETMINREQRECT消息的方法来完成此动作。

```
BOOL MonthCal_GetMinReqRect (
    HWND hwndMC ,
    LPRECT lpRectInfo
);
```

参数

hwndMC: 月历控件的句柄。

lpRectInfo: RECT数据结构地址, 这个数据结构可以接收约束矩形的信息。这个参数必须是一个合法的地址, 并且不能为NULL。

返回值

如果操作成功, 则返回非零值, 并且lpRectInfo将接收可应用的约束矩形信息。否则, 将返回0。

说明

月历控件所需的最小窗口大小取决于当前选择的字体、控件样式、系统度量以及区域设置。当应用程序对上述任何信息进行了改变时, 都将会影响到最小窗口尺寸, 或者会处理WM_SETTINGCHANGE消息, 它将调用MonthCal_GetMinReqRect函数来决定新的最小尺寸。

注意 由MonthCal_GetMinReqRect所返回的矩形并不包括“今天”字符串的宽度。如果MCS_NOTODAY样式没有被设置, 那么应用程序可以调用MonthCal_GetMaxTodayWidth宏来获取定义“经验”字符串宽度的矩形信息。在实际使用时, 应该使用这两个矩形中较大的一个, 从而确保“今天”字符串不会被截取。

lpRectInfo的top数据成员与left数据成员总是为0。right与bottom数据成员分别表示控件所需的最小cx与cy。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetMonthDelta

获取月历控件的滚动频率, 所谓滚动频率是当用户单击滚动按钮时控件转移显示时所移动的月份数。可以使用这个宏, 也可以通过显式发送MCM_GETMONTHDELTA消息的方法来完成此动作。

```
INT MonthCal_GetMonthDelta (
    HWND hwndMC
);
```

参数

hwndMC: 月历控件的句柄。

返回值

如果先前已经使用MonthCal_SetMonthDelta宏设置了月份增量, 那么将返回代表月历当前滚动频率的INT值。如果先前没有使用MonthCal_SetMonthDelta宏设置月份增量, 或者月份增量被重新设置为缺省值, 那么将返回用来表示当前可见月份数的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetMonthRange

使用SYSTEMTIME数据结构, 获取用来表示月历控件所应该显示的最高限制与最低限制的数据信息。可以使用这个宏, 也可以通过显式发送MCM_GETMONTHRANGE消息的方法来完成此动作。

```
DWORD MonthCal_GetMonthRange (
    HWND hwndMC,
    DWORD dwFlag,
    LPSYSTEMTIME lprgSysTimeArray
);
```

参数

hwndMC: 月历控件的句柄。

dwFlag: 用来表示将要被获取的范围限制的值, 这个值必须是以下之一:

GMR_DAYSTATE 包括那些仅仅部分可见的第一月份范围与结尾月份范围。

GMR_VISIBLE 仅仅包括那些完全被显示出来的月份。

lprgSysTimeArray: SYSTEMTIME数据结构数组（两个元素）的地址，这个数组可以获取由dwFlag所指定的最低限制范围与最高限制范围。其中，最低限制范围与最高限制范围分别放置在lprgSysTimeArray[0]与lprgSysTimeArray[1]中。这些数据结构的时间数据成员并不会被修改，同时，这个参数必须是一个合法的地址，而且不能为NULL。

返回值

返回一个INT值，这个INT值表示在lprgSysTimeArray中所指定的范围之间的月份数。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

MonthCal_GetRange

获取月历控件的最小与最大允许日期集合。可以使用这个宏，也可以通过显式发送MCM_GETRANGE消息的方法来完成此动作。

```
DWORD MonthCal_GetRange (
    HWND hwndMC,
    LPSYSTEMTIME lprgSysTimeArray
);
```

参数

hwndMC: 月历控件的句柄。

lprgSysTimeArray: SYSTEMTIME数据结构数组（两个元素）的地址，这个数据结构可以接收日期限制信息。其中，最小日期限制放置在lprgSysTimeArray[0]中，最大日期限制则放置在lprgSysTimeArray[1]中。如果其中有某个元素被设置为全0，那么月历控件的相应部分就没有任何限制。这些数据结构的时间数据成员将不会被修改，并且这个参数必须是一个合法的、不为NULL的地址。

返回值

返回一个DWORD值，这个DWORD值可以是0（表示没有任何限制），或者是下列用来指定限制信息的组合：

GDTR_MAX	表示月历控件没有最大日期限制；同时，lprgSysTimeArray[0]合法，并且包含可应用的日期信息。
GDTR_MIN	表示月历控件没有最小日期限制；同时，lprgSysTimeArray[1]合法，并且包含可应用的日期信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetSelRange

获取当前用户所选择的日期单位中表示最大日期限制与最小日期限制的日期信息。可以使用这个宏, 也可以通过显式发送MCM_GETSELRANGE消息的方法来完成此动作。

```
BOOL MonthCal_GetSelRange (
    HWND hwndMC,
    LPSYSTEMTIME lpSysTimeArray
);
```

参数

hwndMC: 月历控件的句柄。

lpSysTimeArray: SYSTEMTIME数据结构数组 (两个元素) 的地址, 这个数组将获取用户所选择的最小日期限制与最大日期限制。其中, 最小日期限制与最大日期限制分别放置在lpSysTimeArray[0]与lpSysTimeArray[1]中。这些数据结构的时间数据成员不会被修改, 并且这个参数必须是一个合法的、不是NULL的地址。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。如果这个MonthCal_GetSelRange宏应用于没有使用MCS_MULTISELECT样式的月历控件, 那么对宏的使用将导致失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetToday

获取月历控件中被指定为“今天”的日期信息。可以使用这个宏, 也可以通过显式发送MCM_GETTODAY消息的方法来完成此动作。

```
BOOL MonthCal_GetToday (  
    HWND hwndMC,  
    LPSYSTEMTIME lpToday  
);
```

参数

hwndMC: 月历控件的句柄。

lpToday: SYSTEMTIME数据结构的地址, 这个数据结构可以接收日期信息。此外, 这个数据结构的时间数据成员将不会被修改, 而且本参数必须是一个合法的非NULL地址。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_GetUnicodeFormat

获取月历控件的UNICODE字符格式标志。可以使用这个宏, 也可以通过显式发送MCM_GETUNICODEFORMAT消息的方法来完成此动作。

```
BOOL MonthCal_GetUnicodeFormat (  
    HWND hwnd  
);
```

参数

hwnd: 月历控件的句柄。

返回值

返回月历控件的UNICODE格式标志。如果这个值非零, 那么表示控件正在使用UNICODE字符。如果这个值为0, 那么表示控件正在使用ANSI字符。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

MonthCal_SetUnicodeFormat。

MonthCal_HitTest

判断月历控件的哪个部分正在屏幕的给定点上。可以使用这个宏，也可以通过显式发送MCM_HITTEXT消息的方法来完成此动作。

```
DWORD MonthCal_HitTest (
    HWND hwndMC,
    PMCHITTESTINFO pMHitTest
);
```

参数

hwndMC: 月历控件的句柄。

pMHitTest: MCHITTESTINFO数据结构的地址。在调用这个宏时，cbSize数据成员必须被设置为MCHITTESTINFO数据结构的大小，pt数据成员必须被设置为将要进行鼠标点击中测试的点。

返回值

设置由pMHitTest所指向的MCHITTESTINFO数据结构中数据成员的值，并且返回一个包含鼠标点击中测试结果标志集合的DWORD值。关于鼠标点击中测试结果标志的列表，请参见MCM_HITTEST的返回值描述。

pMHitTest所指向的MCHITTESTINFO数据结构的uHit数据成员将等于返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetColor

为月历控件的给定部分设置颜色。可以使用这个宏，也可以通过显式发送MCM_SETCOLOR消息的方法来完成此动作。

```
COLORREF MonthCal_SetColor (
    HWND hwndMC,
    INT iColor,
    COLORREF clr
);
```

参数

hwndMC: 月历控件的句柄。

iColor: 指定将要设置哪个月历控件颜色的INT值, 这个值必须是下列之一:

MCSC_BACKGROUND	获取显示在月份之间的背景色。
MCSC_MONTHBK	获取显示在月份内部的背景色。
MCSC_TEXT	获取显示在月份内部的文本颜色。
MCSC_TITLEBK	获取显示在月历标题中的背景色。
MCSC_TITLETEXT	获取月历控件标题中显示的文本颜色。
MCSC_TRAILINGTEXT	获取用来显示第一日文本与最后一日文本的颜色。第一日与最后一日是指当前月历控件中所出现的上一个月份与下一个月份的开始一天与最后一天。

clr: 表示将要为月历控件的指定区域设置的颜色的COLORREF值。

返回值

如果操作成功, 则返回表示先前为月历控件设置的颜色的COLORREF值; 否则, 返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetCurSel

设置为月历控件所选择的当前日期。如果所指定的日期当前不在视图中, 控件将更新显示以便使得这个日期处于视图中。可以使用这个宏, 也可以通过显式发送MCM_SETCURSEL消息的方法来完成此动作。

```
BOOL MonthCal_SetCurSel (
    HWND hwndMC,
    LPSYSTEMTIME lpSysTime
);
```

参数

hwndMC: 月历控件的句柄。

lpSysTime: SYSTEMTIME数据结构的地址, 在这个数据结构中包含有将要被设置作为当前选择日期的信息。这个数据结构的时间数据成员将被忽略。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。如果这个控件应用于被设置为MCS_MULTISELECT样式的月历控件, 那么将导致失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetDayState

为月历控件中当前可见的所有月份设置日状态。可以使用这个宏, 也可以通过发送MCM_SETDAYSTATE消息的方法来完成此动作。

```
BOOL MonthCal_SetDayState (
    HWND hwndMC,
    INT iMonths,
    LPMONTHDAYSTATE lpDayStateArray
);
```

参数

hwndMC: 月历控件的句柄。

iMonths: 用来表示有多少个元素在lpDayStateArray所指向的数组中的INT值。

lpDayStateArray: MONTHDAYSTATE值数组的地址, 这个数组用来定义月历控件将如何在它的显示区中绘制月历控件。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

说明

lpDayStateArray所指向的数组必须包含足够多的元素, 从而能够容纳由下列宏所返回的所有值:

```
MonthCal_GetMonthRange (hwndMC, GMR_DAYSTATE, NULL);
```

上面的这个宏将返回在月历控件的显示中完全在视图范围之内或者部分在视图范围之内的月份总数。

需要注意的是, 在lpDayStateArray数组中必须包含与控件中当前显示的所有月份相对应的MONTHDAYSTATE值 (按照年历顺序)。这些月份还包括被部分显示的第一个月以及最后一个月。关于这个数组的更多信息, 请参见18.2.3节“准备MONTHDAYSTATE数组”。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的

Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetFirstDayOfWeek

为月历控件设置星期的第一天。可以使用这个宏, 也可以通过显式发送MCM_SETFIRSTDAYOFWEEK消息的方法来完成此动作。

```
DWORD MonthCal_SetFirstDayOfWeek (
    HWND hwndMC,
    INT iDay
);
```

参数

hwndMC: 月历控件的句柄。

iDay: 用来表示哪一天将被设置为星期中第一天的INT值, 这个值必须是一个日编号。

返回值

返回一个包含有两个值的DWORD值。如果先前的星期中第一天设置不等于LOCALE_IFIRSTDAYOFWEEK, 那么DWORD值的高位字部分将是一个非零的BOOL值, 否则为0。DWORD值的低位字部分是一个代表星期中第一天的INT值。

说明

如果星期中第一天被设置为不是缺省值LOCALE_IFIRSTDAYOFWEEK的任何其他值, 那么控件将根据所做的改变自动地更新星期中第一天信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetMaxSelCount

设置可以在月历控件中选择的最大天数。可以使用这个宏, 也可以通过显式发送MCM_SETMAXSELCOUNT消息的方法来完成此动作。

```
BOOL MonthCal_SetMaxSelCount (
    HWND hwndMC,
    UINT iMax
);
```

参数

hwndMC: 月历控件的句柄。

iMax: 将要被设置为表示可被选择的最大总天数的INT值。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。如果这个宏应用于没有使用MCS_MULTISELECT样式的月历控件, 那么将导致失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetMonthDelta

为月历控件设置滚动频率, 所谓滚动频率是指当用户单击某个滚动按钮时, 控件将转移的月份数目。可以使用这个宏, 也可以通过显式发送MCM_SETMONTHDELTA消息的方法来完成此动作。

```
INT MonthCal_SetMonthDelta (
    HWND hwndMC,
    INT iDelta
);
```

参数

hwndMC: 月历控件的句柄。

iDelta: 表示将要被设置作为月历控件滚动频率的月份数值。如果这个值为0, 月历增量将被重新设置为缺省值 (显示在控件中的月份数目)。

返回值

返回一个表示先前滚动频率的INT值。如果先前没有设置滚动频率, 那么这个返回值将为0。

说明

PAGE UP键与PAGE DOWN键, 以及VK_PRIOR按钮与VK_NEXT按钮将改变所选择的月份一次, 而不管所显示的月份数目或者由MCM_SETMONTHDELTA所设置的值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

MonthCal_SetRange

为月历控件设置最小与最大的允许日期。可以使用这个宏，也可以通过显式发送MCM_SETRANGE消息的方法来完成此动作。

```
BOOL MonthCal_SetRange (
    HWND hwndMC,
    DWORD fWhichLimit,
    LPSYSTEMTIME lprgSysTimeArray
);
```

参数

hwndMC：月历控件的句柄。

fWhichLimit：用来指定哪个日期限制正在被设置的值，这个值必须是下面值的一个或两个：

GDTR_MAX 将要被设置的最大允许日期。lprgSysTimeArray[1]上的SYSTEMTIME数据结构必须包含日期信息。

GDTR_MIN 将要被设置的最小允许日期。lprgSysTimeArray[0]上的GDTR_MAX数据结构必须包含日期信息。

lprgSysTimeArray：SYSTEMTIME数据结构数组（二元数）的地址，在这个数组中包含有日期限制信息。其中，如果指定了GDTR_MAX，那么lprgSysTimeArray[1]中将包含最大日期限制；如果指定了GDTR_MIN，那么lprgSysTimeArray[0]中将包含最小日期限制。

返回值

若操作成功，则返回一个非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

MonthCal_SetSelRange

将月历控件中的选中日期设置给一个给定的日期范围。可以使用这个宏，也可以通过显式发送MCM_SETSELRANGE消息的方法来完成此动作。

```
BOOL MonthCal_SetSelRange (
    HWND hwndMC,
    LPSYSTEMTIME lprgSysTimeArray
);
```

参数

hwndMC: 月历控件的句柄。

lpSysTimeArray: SYSTEMTIME数据结构数组(两个元素)的地址, 这个数组中包含有代表选中日期范围的日期信息。第一个所选择的日期必须在lpSysTimeArray[0]中指定, 最后一个被选择日期必须在lpSysTimeArray[1]中指定。这些数据结构的时间数据成员将被忽略。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。如果这个宏应用于没有被设置为MCS_MULTISELECT样式的月历控件, 那么将导致失败。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetToday

为月历控件设置“今天”选择。可以使用这个宏, 也可以通过显式发送MCM_SETTODAY消息的方法来完成此动作。

```
void MonthCal_SetToday (
    HWND hwndMC,
    LPSYSTEMTIME lpSysTime
);
```

参数

hwndMC: 月历控件的句柄。

lpSysTime: SYSTEMTIME数据结构的地址, 在这个数据结构中包含有将要被设置作为控件中“今天”选择的日期信息。如果这个参数被设置为NULL, 那么控件将返回缺省的设置。这个数据结构中的时间数据成员将被忽略。

如果“今天”选择被设置为不同于缺省值的任何日期, 那么以下情况将适用:

- 如果时间已经超过了当天的午夜, 控件将不会自动地更新“今天”选择。
- 控件将不会基于所做的改变来自动地更新自己的显示。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

MonthCal_SetUnicodeFormat

设置月历控件的UNICODE字符格式标志。这个消息允许在运行时改变由月历控件所使用的字符集合, 而不必重新创建控件。可以使用这个宏, 也可以通过显式发送MCM_SETUNICODEFORMAT消息的方法来完成此动作。

```
BOOL MonthCal_SetUnicodeFormat (
    HWND hwnd,
    BOOL fUnicode
);
```

参数

hwnd: 控件的句柄。

fUnicode: 判断由控件所使用的字符集合。如果这个值非零, 那么控件将使用UNICODE字符。如果这个值为0, 那么控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式标志。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

MonthCal_GetUnicodeFormat。

18.5.3 月历控件通告消息

MCN_GETDAYSTATE

为了获取关于单独的天将如何被显示的信息, 月历控件将发送这个通告消息。这个通告消息只能由使用MCS_DAYSTATE样式的月历控件来发送, 并且是以WM_NOTIFY消息的形式进行发送的。

```
MCN_GETDAYSTATE
    lpNMDayState = (LPNMDAYSTATE) lParam;
```

参数

lpNMDayState: NMDAYSTATE数据结构的地址。在这个数据结构中包含有关于哪个控件需要信息的时间帧,并且能够接收提供这个数据的数组地址。

说明

对这个通告消息的处理将使得可以指定某些天被显示为黑体,从而允许应用程序自定义显示方式。关于处理这个通告消息的更多信息,请参见18.2.2节“处理MCN_GETDAYSTATE通告消息”。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCN_SELCHANGE

在当前所选择的日期或者日期范围发生改变时,月历控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
MCN_SELCHANGE
lpNMSELChange = (LPNMSELCHANGE) lParam;
```

参数

lpNMSELChange: NMSELCHANGE数据结构的地址,在这个数据结构中包含有关于当前所选择的日期范围的信息。

说明

例如,当用户在当前月份中显式地改变了自己的选择,或者在响应前/后月份浏览时隐含地改变了选择日期时,控件将发送MCN_SELCHANGE通告消息。

这个通告消息与MCN_SELECT类似,但它是作为对任何日期选择改变的响应而被发送的。MCN_SELECT消息仅仅为某个特定的日期选择发送。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

MCN_SELECT

当用户在月历控件中明确地进行了日期选择时，月历控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
MCN_SELECT  
    lpNMSELChange = (LPNMSELCHANGE) lParam;
```

参数

lpNMSELChange: NMSELCHANGE数据结构的地址，在这个数据结构中包含有关于当前选择的日期范围的信息。

说明

这个通告消息类似于MCN_SELCHANGE通告消息，但它是作为对用户明确选择的日期响应而发送的，MCN_SELCHANGE通告消息则可以适用于任何日期选择改变。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NM_RELEASEDCAPTURE (mohthcal)

这个通告消息将告知月历控件的父窗口，控件将要释放鼠标捕获的信息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE  
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

18.5.4 月历控件数据结构

MCHITTESTINFO

这个数据结构中包含有月历控件的鼠标击中测试点信息，它将与MCM_HITTEST消息以及相应的MonthCal_HitTest宏一同使用。

```
typedef struct {
    UINT      cbSize;
    POINT     pt;
    UINT      uHit;        //An output member
    SYSTEMTIME st;         //An output member
}MCHITTESTINFO,*PMCHITTESTINFO;
```

成员

cbSize: 这个数据结构的大小，单位为字节。

pt: 包含有将要被进行鼠标击中测试的点的POINT数据结构。

uHit: 输出数据成员，这个数据成员可以接受用来表示鼠标击中测试操作结果的位标志。这个值应该是下列值之一：

MCHT_CALEDARBK

MCHT_CALEDARDATE

MCHT_CLAENDARDATENEXT

MCHT_CALEDARDATEPREV

MCHT_CALEDARDAY

MCHT_CALEDARWEEKNUM

MCHT_NOWHERE

MCHT_TITLEBK

表示给定的点在月历的背景中。

表示给定的点在月历的某个特定日期上。lpCHitTest->st上的SYSTEMTIME数据结构将被设置为给定点上的这个日期。

表示给定的点在下一个月份的某个日期上（这个月份被部分地显示在当前显示月份的末端）。如果用户单击这个位置，那么月历控件将滚动显示到下一个月份或者月份集合中。

表示给定的点在上一个月份的某个日期上（这个月份被部分地显示在当前显示月份的末端）。如果用户单击这个位置，那么月历控件将滚动显示到上一个月份或者月份集合中。

表示给定的点在日期缩写上（例如Fri）。这时，lpCHitTest->st上的SYSTEMTIME数据结构将被设置为顶部行的相应日期。

表示给定的点在星期编号上（仅仅对MCS_WEEKNUMBERS样式有效）。这时，lpMCHitTest->st上的SYSTEMTIME数据结构将被设置为最左端列上的相应日期。

表示给定的点不在月历控件上，或者表示给定的点在月历控件的非活跃部分。

表示给定的点在月份标题的背景上。

MCHT_TITLEBTNNEXT

表示给定的点在控件右上角的按钮上。如果用户单击了这个点,那么月历控件将滚动显示到下一个月份或者月份集合中。

MCHT_TITLEBTNPREV

表示给定的点在控件左上角的按钮上。如果用户单击了这个点,那么月历控件将滚动显示到上一个月份或者月份集合中。

MCHT_TITLEMONTH

表示给定的点在月份标题条的一个月份名上。

MCHT_TITLEYEAR

表示给定的点在月份标题条的一个年份值上。

st: SYSTEMTIME数据结构,这个数据结构可以接收被进行鼠标击中测试位置的日期与时间信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMDAYSTATE

这个数据结构中包含有在处理MCN_GETDAYSTATE通告消息时所需要的信息。在处理MCN_GETDAYSTATE通告消息时,除了prgDayState数据成员是用来接收应用程序必须设置的信息之外,本数据结构中的所有数据成员都是用于进行输入的。

```
typedef struct tagNMDAYSTATE {
    NMHDR          nmhdr;
    SYSTEMTIME      stStart;
    int             cDayState;
    LPMONTHDAYSTATE prgDayState;
} NMDAYSTATE, FAR * LPNMDAYSTATE;
```

成员

nmhdr: NMHDR数据结构的地址,在这个数据结构中包含有关于本通告消息的更多信息。

stStart: SYSTEMTIME数据结构,这个数据结构包含有开始日期。

cDayState: 用来指定prgDayState数组中必须有的总元素数的INT值。

prgDayState: MONTHDAYSTATE值数组的地址。这个数组的缓冲器必须足够大,从而足以容纳cDayState中的元素。这个数组中的第一个元素必须与stStart中的日期相对应。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMSELCHANGE

在这个数据结构中包含有处理MCN_SELCHANGE通告消息所需要的信息。

```
typedef struct tagNMSELCHANGE{
    NMHDR        nmhdr;
    SYSTEMTIME   stSelStart;
    SYSTEMTIME   stSelEnd;
}NMSELCHANGE, FAR * LPNMSELCHANGE;
```

成员

nmhdr: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

stSelStart: SYSTEMTIME数据结构, 在这个数据结构中包含有用户所选择的日期范围的第一天。

stSelEnd: SYSTEMTIME数据结构, 在这个数据结构中包含有用户所选择的日期范围的最后一天。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

18.5.5 月历控件数据类型

MONTHDAYSTATE

下面是在Commctrl.h头文件中所定义的新数据类型:

```
typedef DWORD MONTHDAYSTATE, FAR * LPMONTHDAYSTATE;
```

MONTHDAYSTATE数据类型是一种位域, 其中每个位(1~31)分别表示一个月中的日状态。如果这个位被打开, 那么相应的天将被显示为黑体; 否则, 这个天将不被着重显示。

这个数据类型应该与MCM_SETDAYSTATE消息以及相应的MonthCal_SetDayState宏一同使用。当使用MONTHDAYSTATE值时, 就可以用来对少于31天的月份进行处理, 只有那些必要的位才能被访问。

第19章 Pager 控件

Pager控件是一个窗口容器，当某个窗口没有足够的显示空间来显示其所有的内容时，就可以使用Pager控件。Pager控件允许用户对窗口区域进行滚动，从而显示那些不在当前视图范围内的内容。

19.1 关于Pager控件

Microsoft Internet Explorer 4.0 (commctrl.dll 4.71版本)中引入了Pager控件，在某个窗口没有足够的区域来显示一个子窗口时，Pager控件就非常有用。例如，如果应用程序中有一个没有足够的宽度来显示其中所有项目的工具条，就可以将这个工具条分配给一个Pager控件，然后，用户就可以向左或向右滚动，从而获取工具条中的所有项目。此外，也可以创建在竖直方向上进行滚动的Pager控件。

被分配给Pager控件的窗口称为被包含窗口 (contained window)。

图19-1显示了一个包含在Pager控件中的工具条，它就可以用来控制控件中的哪部分区域可见。

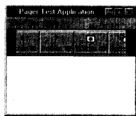


图19-1 Pager控件

注意 Pager控件在Comctl32.dll 4.71版本及其后续版本中实现。

19.2 使用Pager控件

本节讨论如何在应用程序中实现Pager控件。

19.2.1 初始化Pager控件

为了能够使用Pager控件，必须调用InitCommonControlsEx函数，并且在进行函数调用时设置INITCOMMONCONTROLSEX数据结构dwICC数据成员中的ICC_PAGESCROLLER_CLASS标志位集合。

19.2.2 创建Pager控件

可以使用CreateWindow或CreateWindowEx API函数来创建Pager控件，Pager控件的类名是WC_PAGESCROLLER，这个类在Commctrl.h头文件中定义。其中，PGS_HORZ样式用来创建水平的Pager控件，PGS_VERT样式用来创建竖直的Pager控件。由于它是一个子控件，因此还应该使用WS_CHILD样式。

在创建Pager控件之后，就可以考虑向其中分配被包含窗口。如果被包含窗口是一个子窗口，那么就必须要使得这个子窗口是Pager控件的一个子控件，从而保证这个子窗口的大小与位置能够

被正确地计算。然后,就可以使用PGM_SETCCHILD消息将这个窗口分配给Pager控件。需要说明的是,这个消息并不会真正改变被包含窗口的父窗口,而只是分配被包含窗口。如果被包含窗口是一个通用控件,那么这个窗口必须具有CCS_NORESIZE样式,从而防止控件试图改变自己的大小到与Pager控件的大小相同。

19.2.3 处理Pager控件通告消息

一般情况下,应用程序至少应该能够处理PGN_CALCSIZE通告消息。如果不对这个通告消息进行处理,而是为宽度或高度输入一个值,那么Pager控件中的滚动箭头将不会显示出来。这是因为Pager控件需要使用PGN_CALCSIZE通告消息中所提供的宽度或高度值来决定被包含窗口的“理想”大小。

下面这个例子演示了如何处理PGN_CALCSIZE通告消息。在这个例子中,被包含窗口是一个工具条控件,在这个工具条控件中包含有未知数目、未知大小的按钮。这个例子演示了如何使用TB_GETMAXSIZE消息来计算工具条中所有项目的大小。然后,将所有项目的宽度值放置在传递给通告消息的NMPGCALCSIZE数据结构iWidth数据成员中。

```
case PGN_CALCSIZE:
{
    LPNMPGCALCSIZE pCalcSize = (LPNMPGCALCSIZE)lParam;

    switch(pCalcSize->dwFlag)
    {
        case PGF_CALCWIDTH:
        {
            SIZE size;

            //Get the optimum width of the toolbar.
            SendMessage(hwndToolbar,TB_GETMAXSIZE,0,(LPARAM)&size);
            pCalcSize->iWidth =size.cx;
        }
        break;
    }
}

return 0;
```

对PGN_SCROLL通告消息的处理是可选的。如果需要知道何时发生了滚动动作、需要跟踪滚动位置或者需要改变滚动增量,那么就应该处理这个通告消息。如果需要取消滚动功能,那么只要在传递给这个通告消息的NMPGSCROLL数据结构的iScroll数据成员中设置0即可。

下面的例子演示了如何修改滚动增量:

```
case PGN_SCROLL:
{
    LPNMPGSCROLL pScroll = (LPNMPGSCROLL)lParam;

    switch(pScroll->iDir)
    {
```

```

    case PGF_SCROLLLEFT:
    case PGF_SCROLLRIGHT:
    case PGF_SCROLLUP:
    case PGF_SCROLLDOWN:
        pScroll->iScroll +=20;
        break;
    }
}
return 0;

```

19.3 Pager控件样式

在创建Pager控件时，可以使用以下窗口样式：

PGS_AUTOSCROLL	当用户将鼠标放置在某个滚动按钮上时，Pager控件将滚动。
PGS_DRAGNDROP	被包含窗口可以是一个拖放式目标。如果某个项目从Pager外部拖动到某个滚动按钮上，Pager控件将自动滚动。
PGS_HORZ	创建可以进行水平滚动的Pager控件。这个样式与PGS_VERT样式是互斥的，两者不能组合使用。
PGS_VERT	创建可以进行竖直滚动的Pager控件。如果没有特别指定滚动方向样式，那么缺省情况下将使用这个样式。此外，这个样式与PGS_HORZ样式是互斥的，两者不能组合使用。

19.4 Pager控件参考

19.4.1 Pager控件消息

PGM_FORWARDMOUSE

这个消息用来打开或关闭Pager控件的鼠标消息转发功能。当鼠标消息转发功能打开时，Pager控件将把WM_MOUSEMOVE消息转发给被包含窗口。可以显式地发送这个消息，也可以使用Pager_ForwardMouse宏来发送这个消息。

```

PGM_FORWARDMOUSE
wParam = (WPARAM)(BOOL)bForward;
lParam = 0;

```

参数

bForward：用来决定鼠标消息转发功能是打开还是关闭的BOOL值。如果这个值为非零，那么表示鼠标消息转发功能被打开。如果这个值为0，那么表示鼠标消息转发功能被关闭。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PGM_GETBKCOLOR

获取Pager控件的当前背景色。可以显式地发送这个消息, 也可以使用Pager_GetBkColor宏来发送这个消息。

```
PGM_GETBKCOLOR
    wParam = 0;
    lParam = 0;
```

返回值

返回包含当前背景色的COLORREF值。

说明

缺省情况下, Pager控件将使用系统按钮的表面颜色作为背景色。这个颜色与通过调用GetSysColor (利用COLOR_BTNFACE) 所得到的颜色相同。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PGM_GETBORDER

获取Pager控件当前边界大小。可以显式地发送这个消息, 也可以使用Pager_GetBorder宏来发送这个消息。

```
PGM_GETBORDER
    wParam = 0;
    lParam = 0;
```

返回值

返回包含当前边界大小 (单位为像素) 的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版

本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

PGM_SETBORDER。

PGM_GETBUTTONSIZE

获取Pager控件的当前按钮大小。可以显式地发送这个消息, 也可以使用Pager_GetButtonSize宏来发送这个消息。

```
PGM_GETBUTTONSIZE
```

```
wParam = 0;
```

```
lParam = 0;
```

返回值

返回包含当前按钮大小(单位为像素)的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

PGM_SETBUTTONSIZE。

PGM_GETBUTTONSTATE

获取Pager控件中指定按钮的状态。可以显式地发送这个消息, 也可以使用Pager_GetButtonState宏来发送这个消息。

```
PGM_GETBUTTONSTATE
```

```
wParam = 0;
```

```
lParam = (LPARAM)(int)iButton;
```

参数

iButton: 这个参数用来表示需要获取其状态的按钮, 它可以是以下值之一:

PGB_TOPORLEFT

表示PGS_VERT Pager控件中的顶部按钮或者PGS_HORZ Pager控件中的左边按钮。

PGB_BOTTOMORRIGHT

表示PGS_VERT Pager控件中的底部按钮或者PGS_HORZ

Pager控件中的右边按钮。

返回值

返回由iButton参数所指定的按钮的状态，可以是下列状态之一：

PGF_INVISIBLE	表示按钮不可见。
PGF_NORMAL	表示按钮处于普通状态。
PGF_GRAYED	表示按钮处于灰色状态。
PGF_DEPRESSED	表示按钮处于按下状态。
PGF_HOT	表示按钮处于热状态。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PGM_GETDROPTARGET

获取Pager控件的IDropTarget接口指针。可以显式地发送这个消息，也可以使用Pager_GetDropTarget宏来发送这个消息。

```
PGM_GETDROPTARGET
    wParam = 0;
    lParam = (IDropTarget**) ppDropTarget;
```

参数

ppDropTarget：用来获取接口指针的IDropTarget指针。当不再需要这个指针时，调用程序必须用Release函数来释放这个指针。

返回值

这个消息的返回值没有使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PGM_GETPOS

获取Pager控件的当前滚动位置。可以显式地发送这个消息，也可以使用Pager_GetPos宏来发送这个消息。

```
PGM_GETPOS  
    wParam = 0;  
    lParam = 0;
```

返回值

返回包含当前滚动位置（单位为像素）的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PGM_RECALCSIZE

强制Pager控件重新计算被包含窗口大小。在发送这个消息之后，将导致PGN_CALCSIZE通告消息被发送。可以显式地发送这个消息，也可以使用Pager_RecalcSize宏来发送这个消息。

```
PGM_RECALCSIZE  
    wParam = 0;  
    lParam = 0;
```

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PGM_SETBKCOLOR

设置Pager控件的当前背景色。可以显式地发送这个消息，也可以使用Pager_SetBkColor宏来发送这个消息。

```
PGM_SETBKCOLOR
    wParam = 0;
    lParam = (LPARAM)(COLORREF) clrBK;
```

参数

clrBK: 包含有Pager控件新背景色的COLORREF值。

返回值

返回包含Pager控件先前背景色的COLORREF值。

说明

缺省情况下, Pager控件将使用系统按钮的表面颜色作为背景色。这个颜色与调用GetSysColor函数(用COLOR_BTNFACE参数)所获取的颜色相同。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PGM_SETBORDER

设置Pager控件的当前边界大小。可以显式地发送这个消息, 也可以使用Pager_SetBorder宏来发送这个消息。

```
PGM_SETBORDER
    wParam = 0;
    lParam = (LPARAM)(int) iBorder;
```

参数

iBorder: Pager控件边界的新大小(单位为像素), 这个值不应该大于Pager按钮的值, 也不应该小于0。如果iBorder值过大, 那么所画出来的边界将与按钮一样大。如果iBorder值为负, 那么Pager控件的边界大小将被设置为0。

返回值

返回包含先前边界大小的INT值(单位为像素)。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PGM_SETBUTTONSIZE

设置Pager控件的当前按钮大小。可以显式地发送这个消息，也可以使用Pager_SetButtonSize宏来发送这个消息。

```
PGM_SETBUTTONSIZE  
wParam = 0;  
lParam = (LPARAM)(int) iButtonSize;
```

参数

iButtonSize: 包含新按钮大小的INT值（单位为像素）。

返回值

返回包含先前按钮大小的INT值（单位为像素）。

说明

如果Pager控件使用的是PGS_HORZ样式，那么按钮大小将决定Pager按钮的宽度。如果Pager控件使用的是PGS_VERT样式，那么按钮大小将确定Pager按钮的高度。缺省情况下，Pager控件将把它的按钮大小设置为滚动条宽度的3/4。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

PGM_GETBUTTONSIZE。

PGM_SETCHILD

为Pager控件设置被包含窗口。这个消息并不会改变被包含窗口父窗口，而是仅仅将一个窗口句柄分配给Pager控件，从而进行滚动。在多数情况下，被包含窗口都是一个子窗口，这时，被包含窗口也必须是Pager控件的子控件。可以显式地发送这个消息，也可以使用Pager_SetChild宏来发送这个消息。

```
PGM_SETCHILD  
wParam = 0;  
lParam = (LPARAM)(HWND) hwndChild;
```

参数

hwndChild: 将要被包含的窗口句柄。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PGM_SETPOS

设置Pager控件的当前滚动位置。可以显式地发送这个消息, 也可以使用Pager_SetPos宏来发送这个消息。

```
PGM_SETPOS
    wParam = 0;
    lParam = (LPARAM)(int)iPos;
```

参数

iPos: 包含有新滚动位置的INT值, 单位为像素。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

19.4.2 Pager控件宏

Pager_ForwardMouse

打开或关闭Pager控件鼠标消息转发功能。当打开了鼠标消息转发功能时, Pager控件将把WM_MOUSEMOVE消息转发给被包含窗口。可以使用这个宏, 也可以通过显式发送PGM_FORWARDMOUSE消息的方法来完成此动作。

```
VOID Pager_ForwardMouse (
    HWND hwndPager,
    BOOL bForward
```

);

参数

hwndPager: Pager控件句柄。

bForward: 用来判断是否打开了鼠标消息转发功能的BOOL值。如果这个值为非零, 那么表示打开了鼠标消息转发功能。如果这个值为0, 那么表示没有打开鼠标消息转发功能。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_GetBkColor

获取Pager控件等当前背景色。可以使用这个宏, 也可以通过显式发送PGM_GETBKCOLOR消息的方法来完成此动作。

```
COLORREF Pager_GetBkColor (
    HWND hwndPager
);
```

参数

hwndPager: Pager控件的句柄。

返回值

返回包含当前背景色的COLORREF值。

说明

缺省情况下, Pager控件将使用系统按钮的表面颜色作为背景色。这种颜色与调用GetSysColor函数 (使用COLOR_BTNFACE参数) 所得到的颜色相同。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_GetBorder

获取Pager控件的当前边界大小。包含可以使用这个宏，也可以通过显式发送PGM_GETBORDER消息的方法来完成此动作。

```
int Pager_GetBorder (
    HWND hwndPager
);
```

参数

hwndPager: Pager控件的句柄。

返回值

返回包含当前边界大小的INT值，单位为像素。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_GetButtonSize

获取Pager控件的当前按钮大小。可以使用这个宏，也可以通过显式发送PGM_GETBUTTONSIZE消息的方法来完成此动作。

```
int Pager_GetButtonSize (
    HWND hwndPager
);
```

参数

hwndPager: Pager控件的句柄。

返回值

返回包含当前按钮大小的INT值，单位为像素。

参见

Pager_SetButtonSize。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_GetButtonState

获取Pager控件中指定按钮的状态。可以使用这个宏, 也可以通过显式发送PGM_GETBUTTONSTATE消息的方法来完成此动作。

```
DWORD Pager_GetButtonState (  
    HWND hwndPager;  
    int iButton  
);
```

参数

hwndPager: Pager控件的句柄。

iButton: 这个参数来表示将要获取其状态的按钮。关于这个参数的可能值的更多描述, 请参见19.4.1节中PGM_GETBUTTONSTATE对iButton的描述。

返回值

返回在iButton参数中所指定的按钮的状态。关于返回的可能值的更多描述, 请参见19.4.1节中PGM_GETBUTTONSTATE对返回值的描述。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_GetDropTarget

获取Pager控件的IDropTarget接口指针。可以使用这个宏, 也可以通过显式发送PGM_GETDROPTARGET消息的方法来完成此动作。

```
void Pager_GetDropTarget (  
    HWND hwndPager;  
    IDropTarget **ppDropTarget  
);
```

参数

hwndPager: Pager控件的句柄。

ppDropTarget: 将要获取接口指针的IDropTarget指针地址。当不再需要使用这个指针时, 要函数将负责调用Release来释放指针。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_GetPos

获取Pager控件的当前滚动位置。可以使用这个宏, 也可以通过显式发送PGM_GETPOS消息的方法来完成此动作。

```
COLORREF Pager_GetPos (
    HWND hwndPager;
);
```

参数

hwndPager: Pager控件的句柄。

返回值

返回包含当前滚动位置的INT值, 单位为像素。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_RecalcSize

强制Pager控件重新计算被包含窗口大小, 使用这个宏将导致发送PGN_CALCSIZE通告消息。可以使用这个宏, 也可以通过显式发送PGM_RECALCSIZE消息的方法来完成此动作。

```
COLORREF Pager_RecalcSize (
    HWND hwndPager
);
```

参数

hwndPager: Pager控件的句柄。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_SetBkColor

设置Pager控件的当前背景色。可以使用这个宏, 也可以通过显式发送PGM_SETBKCOLOR消息的方法来完成此动作。

```
COLORREF Pager_SetBkColor (
    HWND hwndPager,
    COLORREF clrBk
);
```

参数

hwndPager: Pager控件的句柄。

clrBk: 包含Pager控件新背景色的COLORREF值。

返回值

返回Pager控件先前背景色的COLORREF值。

说明

缺省情况下, Pager控件将使用系统按钮的表面颜色作为背景色。这个颜色与调用GetSysColor函数(使用COLOR_BTNFACE参数)所得到的颜色相同。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_SetBorder

设置Pager控件的当前边界大小。可以使用这个宏, 也可以通过显式发送PGM_SETBORDER消息的方法来完成此动作。

```
INT Pager_SetBorder (
    HWND hwndPager,
    int iBorder
);
```

参数

hwndPager: Pager控件的句柄。

iBorder: 边界的新大小, 单位为像素, 这个值不应该大于Pager控件的大小或小于0。如果iBorder过大, 那么所画出来的边界将与按钮一样大。如果iBorder为负, 那么边界大小被设置为0。

返回值

返回包含先前边界大小的INT值, 单位为像素。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

Pager_SetButtonSize

设置Pager控件的当前按钮大小。可以使用这个宏, 也可以通过显式发送PGM_SETBUTTONSIZE消息的方法来完成此动作。

```
int Pager_SetButtonSize (
    HWND hwndPager,
    int iButtonSize
);
```

参数

hwndPager: Pager控件的句柄。

iButtonSize: 包含新按钮大小INT值, 单位为像素。

返回值

返回包含先前按钮大小的INT值, 单位为像素。

说明

如果Pager控件使用的是PGS_HORZ样式, 那么按钮大小将决定Pager按钮的宽度。如果Pager控件使用的是PGS_VERT样式, 那么按钮大小将决定Pager按钮高度。缺省情况下, 它就控件将把按钮大小设置为滚动条宽度大小的3/4。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

Pager_GetButtonSize。

Pager_SetChild

为Pager控件分配被包含窗口。这个宏并不会改变被包含窗口的父窗口，它只是将一个窗口句柄分配给Pager控件，从而进行滚动。在大多数情况下，被包含窗口将是一个子窗口，这时，被包含窗口应该是Pager控件的子控件。可以使用这个宏，也可以通过显式发送PGM_SETCHILD消息的方法来完成此动作。

```
COLORREF Pager_SetChild (
    HWND hwndPager,
    HWND hwndChild
);
```

参数

hwndPager: Pager控件的句柄。

hwndChild: 将要被包含的窗口句柄。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

Pager_SetPos

为Pager控件设置滚动位置。可以使用这个宏，也可以通过显式发送PGM_SETPOS消息的方法来完成此动作。

```
INT Pager_SetPos (
    HWND hwndPager;
    int iPos
);
```

参数

hwndPager: Pager控件的句柄。

iPos: 包含新滚动位置的INT值, 单位为像素。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

19.4.3 Pager控件通告消息

NM_RELEASEDCAPTURE (pager)

这个通告消息用来告知Pager控件的父窗口, Pager控件已经释放了鼠标捕获信息。NM_RELEASEDCAPTURE通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

所得的返回值被Pager控件忽略。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PGN_CALCSIZE

这个通告消息由Pager控件所发送, 用来获取被包含窗口的可滚动范围。这些可滚动范围将被Pager控件用来决定被包含窗口的可滚动大小。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PGN_CALCSIZE
    lpnmcs = (LPNMPGCALCSIZE) lParam;
```

参数

lpnmcs: NMPGCALCSIZE数据结构的地址, 这个数据结构可以包含以及获取关于本通告消

息的信息。这个数据结构的dwFlag数据成员表示控件的哪个范围被计算。根据dwFlag值的不同,应该将需要计算的范围放置在数据结构的iWidth数据成员或iHeight数据成员中。

返回值

返回值被忽略。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PGN_SCROLL

在被包含窗口进行滚动之前, Pager控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PGN_SCROLL
lpnms = (LPNMPGSCROLL) lParam;
```

参数

lpnms: NMPGSCROLL数据结构的地址, 这个数据结构可以包含与获取关于本通告消息的信息。这个数据结构中的iDir数据成员用来表示滚动的方向; iXpos数据成员与iYpos数据成员中包含有被包含窗口在滚动之前的位置信息; iScroll数据成员中包含有缺省的滚动增量大小, 可以修改这个数据成员, 从而适合自己所需的滚动增量。

返回值

返回值被忽略。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

19.4.4 Pager控件数据结构

NMPGALCSIZE

在这个数据结构中包含有Pager控件用来计算被包含窗口可滚动区域的信息, 这个数据结构

与PGN_CALCFSIZE通告消息一同使用。

```
typedef struct {
    NMHDR hdr;
    DWORD dwFlag;
    int iWidth;
    int iHeight;
}NMPGCALCFSIZE, *LPNMPGCALCFSIZE;
```

成员

hdr: 包含关于本通告消息信息的NMHDR数据结构。

dwFlag: 这个数据成员用来说明哪个范围正在被请求，它必须是以下值之一：

PGF_CALCHEIGHT 被请求可滚动区域的高度，这个高度值必须在从通告消息返回之前被放置在iHeight数据成员中。

PGF_CALCWIDTH 被请求可滚动区域的宽度，这个宽度值必须在从通告消息返回之前被放置在iWidth数据成员中。

iWidth: 接收可滚动区域所需的宽度，单位为像素。

iHeight: 接收可滚动区域所需的高度，单位为像素。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMPGSCROLL

在这个数据结构中包含在滚动被包含窗口时由Pager控件所使用的信息，这个数据结构必须与PGN_SCROLL通告消息一同使用。

```
typedef struct {
    NMHDR hdr;
    BOOL fwKeys;
    RECT rcParent;
    int iDir;
    int iXpos;
    int iYpos;
    int iScroll;
}NMPGSCROLL, *LPNMPGSCROLL;
```

成员

hdr: 包含有关于本通告消息信息的NMHDR数据结构。

fwKeys: 在发生滚动动作时, 处于被按下状态的修改键, 这个数据成员必须是以下一个或多个值:

0	表示没有修改键被按下
PGK_SHIFT	表示SHIFT键被按下
PGK_CONTROL	表示CONTROL键被按下
PGK_MENU	表示ALT键被按下

rcParent: 包含有Pager控件的客户矩形。

iDir: 这个数据成员用来表示滚动所发生的方向, 可以是以下方向之一:

PGF_SCROLLDOWN	表示被包含窗口正在向下滚动
PGF_SCROLLLEFT	表示被包含窗口正在向左滚动
PGF_SCROLLRIGHT	表示被包含窗口曾的向右滚动
PGF_SCROLLUP	表示被包含窗口正在向上滚动

iXpos: 这个数据成员包含有被包含窗口在滚动发生之前的水平滚动位置信息, 单位为像素。

iYpos: 这个数据成员包含有被包含窗口在滚动发生之前的竖直滚动位置信息, 单位为像素。

iScroll: 这个数据成员包含有缺省的滚动增量, 单位为像素。在必要时, 可以修改这个数据成员, 使其包含一个不同的滚动增量。这个值总是为正, 与滚动方向无关。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

第20章 进度条控件

进度条 (process bar) 是一个窗口, 应用程序可利用进度条来表示一个长时间操作过程的操作进度。在这个进度条中, 包含有一个正在逐渐被系统的高亮颜色所填充的矩形, 用来表示操作的进度。图20-1演示了一个位于窗口底部的进度条。

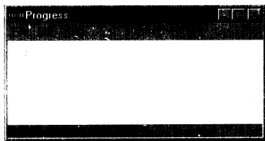


图20-1 进度条

20.1 使用进度条

可以调用 `CreateWindowEx` 函数来创建进度条, 在调用这个函数之前应该声明 `PROGRESS_CLASS` 窗口类, 当通用控件动态链接库 (DLL) 被加载时, 这个窗口类将被注册。为了确保这个 DLL 被加载, 首先应该使用 `InitCommonControls` 函数。

20.1.1 范围与当前位置

进度条的范围表示操作的整个阶段, 而当前位置则表示应用程序完成整个操作所处的进度。窗口进程使用范围与当前位置两个信息来决定将要用高亮颜色进行填充进度条百分比。由于范围与当前位置值是使用无符号整数来进行表示的, 因此范围值以及当前位置值最大不能超过 65 535。

范围的最小值可以在 0~65 535 之间变化。相应地, 它的最大值也可以在 0 到 65 535 之间变化。如果没有设置范围值, 那么系统将把范围的最小值设置为 0, 最大值设置为 100。可以利用 `PBM_SETRANGE` 消息将范围值调整为一个自己所需的整型数。

进度条提供了几个可以用来设置当前位置的消息。其中, `PBM_SETPOS` 消息可用来将当前位置设置为一个给定的值。`PBM_DELTAPOS` 消息可用来将一个给定的值加入到当前位置值, 从而使得当前位置得到推进。

`PBM_SETSTEP` 消息可以用来为进度条指定一个步进增量。这样, 每当向进度条发送 `PBM_STEPIT` 消息时, 进度条的当前位置将向前推进所指定的增量。缺省情况下, 步进增量被设置为 10。

20.1.2 缺省进度条消息处理

本节将描述窗口进程为PROGRESS_CLASS类所处理的消息：

消 息	执行动作
WM_CREATE	分配并初始化一个初始数据结构
WM_DESTROY	释放分配给进度条的所有资源
WM_ERASEBKGD	绘制进度条的背景与边界
WM_GETFONT	返回当前字体的句柄。目前，进度条并不绘制文本，因此发送这个消息对进度条控件不会带来任何影响
WM_PAINT	绘制进度条。如果wParam参数不是NULL，那么控件将假设这个值是一个HDC，并且使用设备上下文进行进度条绘制
WM_SETFONT	保存新字体的句柄，并且返回先前字体的句柄。由于进度条目前并不绘制文本，因此发送这个消息对进度条控件不会带来任何影响

20.1.3 进度条实例

下面的例子演示了如何使用进度条来表示一个长时间文件解析操作的操作进度。这个例子将创建一个进度条，并且在父窗口客户区的底部定位进度条。进度条的宽度是根据滚动条中的箭头位图高度来确定的。进度条的范围是根据文件大小除以2048来确定的（其中，2048是从文件读数据的每个“簇”大小）。此外，在这个例子中还设置了步进增量，并且在解析了每个文件簇之后推进进度条的当前位置一个步进增量单位。

```
//ParseALargeFile -parses a large file and uses a progress bar to
// indicate the progress of the parsing operation.
//Returns TRUE if successful, or FALSE otherwise.
//hwndParent -parent window of the progress bar.
//lpzFileName -name of the file to parse.
//
//Global variable
// g_hinst -instance handle
extern HINSTANCE g_hinst;

BOOL ParseALargeFile(HWND hwndParent, LPSTR lpzFileName)
{
    RECT rcClient; //client area of parent window
    int cyVScroll; //height of scroll bar arrow
    HWND hwndPB; //handle of progress bar
    HANDLE hFile; //handle of file
    DWORD cb; //size of file and count of bytes read
    LPCH pch; //address of data read from file
    LPCH pchTmp; //temporary pointer

    //Ensure that the common control DLL is loaded and create a
    //progress bar along the bottom of the client area of the
```

```

//parent window.Base the height of the progress bar on
//the height of a scroll bar arrow.
InitCommonControls();
GetClientRect(hwndParent,&rcClient);
cyVScroll =GetSystemMetrics(SM_CYVSCROLL);
hwndPB =CreateWindowEx(0,PROGRESS_CLASS,(LPSTR)NULL,
    WS_CHILD |WS_VISIBLE,rcClient.left,
    rcClient.bottom -cyVScroll,
    rcClient.right,cyVScroll,
    hwndParent,(HMENU)0,g_hinst,NULL);

//Open the file for reading,and retrieve the size of the file.
hFile =CreateFile(lpszFileName,GENERIC_READ,FILE_SHARE_READ,
    (LPSECURITY_ATTRIBUTES)NULL,OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,(HANDLE)NULL);

if (hFile ==(HANDLE)INVALID_HANDLE_VALUE)
    return FALSE;

cb =GetFileSize(hFile,(LPDWORD)NULL);

//Set the range and increment of the progress bar.
SendMessage(hwndPB,PBM_SETRANGE,0,MAKELPARAM(0,cb /2048));
SendMessage(hwndPB,PBM_SETSTEP,(WPARAM)1,0);

//Parse the file.
pch =(LPCH)LocalAlloc(LPTR,sizeof(char)*2048);
pchTmp =pch;
do {
    ReadFile(hFile,pchTmp,sizeof(char)*2048,&cb,
        (LPOVERLAPPED)NULL);

    //Include here any code that parses the file.

    //Advance the current position of the progress bar
    //by the increment.
    SendMessage(hwndPB,PBM_STEPIT,0,0);
}while (cb);
CloseHandle((HANDLE)hFile);

DestroyWindow(hwndPB);
return TRUE;
}

```

20.2 在Internet Explorer中更新进度条控件

Microsoft Internet Explorer中的进度条控件支持以下新特性:

新的进度条控件样式

利用PBS_VERTICAL样式，进度条控件现在可以竖直地显示进度信息。此外，还可以使用PBS_SMOOTH样式来支持平滑的进度显示模式。利用PBS_SMOOTH样式，将会导致控件显示一个连续的进度条，而不是分段进度条。

扩展的范围值

进度条控件现在可以支持32位范围值。如果需要设置超过65 535的范围值，就应该使用PBM_SETRANGE32消息。如果需要获取32位的范围值，就应该使用PBM_GETRANGE消息。进度条的范围值上限、下限以及位置参数都是带符号的整型值。为了利用32位的范围值，应该将范围值设置在-0x7FFFFFFF与0x7FFFFFFF之间，并且将位置值当成带符号整数进行处理。

可编程的颜色

应用程序现在可以使用PBM_SETBARCOLOR与PBM_SETBKCOLOR消息来控制进度条控件中所使用的颜色。

20.3 进度条控件样式

进度条控件现在可以支持控件样式，使用者可以像在其他通用控件中一样设置进度条的样式（利用CreateWindowEx，GetWindowLong与SetWindowLong）。下面列出了进度条控件可以支持的样式：

PBS_SMOOTH	在4.70版本中有效。在这种样式上，进度条将以平滑滚动条（而不是缺省的分段进度条）的形式来显示进度状态。
PBS_VERTICAL	在4.70版本中有效。进度条将以竖直的方式从底部到顶部显示进度状态。

20.4 进度条控件参考

20.4.1 进度条控件消息

PBM_DELTAPOS

将进度条的当前位置推进指定的步进增量，然后重绘进度条以反映新的进度条位置。

```
PBM_DELTAPOS
wParam = (WPARAM) nIncrement
lParam = 0;
```

参数

nIncrement：推进进度条位置的步进增量值。

返回值

返回先前位置值。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_GETPOS

获取进度条的当前位置值。

```
PBM_GETPOS
    wParam = 0;
    lParam = 0;
```

返回值

返回代表进度条当前位置的UINT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_GETRANGE

获取给定进度条控件当前范围值上限与下限的信息。

```
PBM_GETRANGE
    wParam = (WPARAM)(BOOL) fWhichLimit;
    lParam = (LPARAM)(PBRANGE) ppBRange;
```

参数

fWhichLimit: 用来说明哪个范围限制值将被用做消息返回值的标志, 这个参数可以是下列值之一:

TRUE 返回下限值

FALSE 返回上限值

ppBRange: PBRANGE数据结构的地址, 这个数据结构将被进度条条件的上限值与下限值所填充。如果这个参数被设置为NULL, 那么进度条控件将只返回由fWhichLimit所指定的范围值。

返回值

返回由fWhichLimit所指定的代表范围值的INT值。如果lParam不是NULL, 那么lParam必须指向将要被上限值与下限值所填充的PBRANGE数据结构。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_SETBARCOLOR

设置进度条控件中进度条指示条的颜色。

```
PBM_SETBARCOLOR  
wParam = 0;  
lParam = (LPARAM)(COLORREF) clrBar;
```

参数

clrBar: 用来指定新进度指示条颜色的COLORREF值。如果为这个参数指定了CLR_DEFAULT值, 那么将导致进度条使用缺省的进度指示条颜色。

返回值

返回进度指示条的先前颜色, 如果进度指示条颜色被设置为缺省颜色, 则返回CLR_DEFAULT。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PBM_SETBKCOLOR

设置进度条的背景色。

```
PBM_SETBKCOLOR  
wParam = 0;  
lParam = (LPARAM)(COLORREF) clrBk;
```

参数

clrBk: 用来表示新背景色的COLORREF值。如果这个参数的值为CLR_DEFAULT, 那么将导致进度条使用缺省的背景色。

返回值

返回先前背景色, 如果背景色为缺省颜色, 则返回CLR_DEFAULT。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PBM_SETPOS

设置进度条的当前位置, 并且重绘进度条以反映进度条的新位置。

```
PBM_SETPOS
    wParam = (WPARAM) nNewPos;
    lParam = 0;
```

参数

nNewPos: 进度条新位置的带符号整数。

返回值

返回先前位置值。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_SETRANGE

设置进度条的最小值与最大值, 并且重绘进度条以反映新的范围值。

```
PBM_SETRANGE
    wParam = 0;
    lParam = MAKELPARAM (nMinRange, nMaxRange);
```

参数

nMinRange: 最小范围值。缺省情况下, 最小范围值为0。

nMaxRange: 最大范围值。缺省情况下, 最大返回值为100。

返回值

如果操作成功, 则返回先前范围值; 否则, 返回0。其中, 返回值的低位字部分用来表示先前的最小值, 高位字部分用来表示先前的最大值。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_SETRANGE32

将进度条控件的范围值设置为一个32位值。

```
PBM_SETRANGE32
wParam = (WPARAM)(int) iLowLim;
lParam = (LPARAM)(int) iHighLim;
```

参数

iLowLim: 用来表示将要为进度条控件设置的下限的带符号整数。

iHighLim: 用来表示将要为进度条控件设置的上限的带符号整数。

返回值

返回一个DWORD值, 这个DWORD值的低位字部分存放有先前的16位下限值, 高位字部分存放有先前的16为上限值。如果先前范围值是32位值, 那么所得到的返回值将包含这两个32位范围限制值的低位字部分。

说明

如果需要获取整个32位上限值与下限值, 则应该使用PBM_GETRANGE消息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_SETSTEP

设置进度条的步进增量。所谓步进增量, 是指每当进度条接收到一个PBM_STEPIT消息时, 进度条对当前位置进行增加的增加数。缺省情况下, 步进增量为10。

```
PBM_SETSTEP
wParam = (WPARAM) nStepInc;
lParam = 0;
```

参数

nStepInc: 新的步进增量。

返回值

返回先前步进增量。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

PBM_STEPIT

向前推进进度条的当前位置一个步进增量单位, 然后重绘进度条以反映新位置。应用程序可以发送PBM_SETSTEP消息来设置步进增量。

```
PBM_STEPIT
    wParam = 0;
    lParam = 0;
```

返回值

返回先前位置。

说明

当位置值超出了最大范围值时, 这个消息将重新设置当前位置, 从而使得进度指示条能够重新从开始位置进行步进。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

20.4.2 进度条控件数据结构**PBRANGE**

在这个数据结构中包含有关于进度条控件上限与下限的信息。这个数据结构必须与PBM_GETRANGE消息一同使用。

```
typedef struct {
    int iLow;
    int iHigh;
} PBRANGE, *PPBRANGE;
```

成员

iLow: 进度条控件的下限值。这个值为带符号整数。

iHigh: 进度条控件的上限值。这个值为带符号整数。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

第21章 属 性 页

属性页 (property sheet) 是一个窗口, 这个窗口允许用户查看并编辑某个项目的属性。例如, 电子表格应用程序就可以使用属性页来允许用户设置某个单元格的字体与边界属性、查看并设置某个设备 (例如磁盘驱动器、打印机、鼠标) 的属性。

21.1 关于属性页

在本章的讨论中, 将假设读者对对话框模板与对话框进程有充分的了解。如果对这些内容还不熟悉, 那么在阅读本章内容之前, 建议先阅读“Platform SDK”(SDK平台)中的“Dialog Boxes”一章。

如果需要在应用程序中实现属性页, 那么就应该将Prsht.h头文件包含在应用程序的工程中。Prsht.h头文件中包含有属性页中所使用的所有标识符。

一个属性页包含有一个或多个相互重叠的子窗口, 这些子窗口成为页面, 在每个子窗口中都包含有用来设置一组相关属性的控件窗口。例如, 在一个页面中可能包含设置项目字体属性的控件 (包括类型样式、点大小、颜色等等)。每个页面都有一个Tab, 用来将页面引入到属性页的前端。例如, 日期/时间控件面板 (Date/Time Control Panel) 应用程序将显示如图21-1所示的属性页。

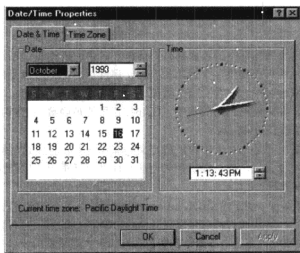


图21-1 Date/Time属性页

有一类名为向导的特殊属性页, 向导可用来按照应用程序控制的序列在某个特定的时刻显示页面。在向导中, 不是通过单击Tab的方法来从一组页面中进行页面选择, 而是通过单击位于

向导底部的Next（下一步）或Back（上一步）按钮来按照某个特定的序列对页面进行向前与向后的导航。例如，图21-2给出了Hardware控制面板应用程序向导中的欢迎页面。



图21-2 Hardware Wizard欢迎界面

关于向导的更多讨论，请参见第11章。

21.1.1 属性页对话框

事实上，属性页以及属性页中的页面都是对话框。其中，属性页是系统定义的对话框，它用来管理页面并且为这些页面提供一个通用的容器。属性页对话框可以是模态对话框，也可以是非模态对话框。在属性页中，包含有一个框架、标题条以及四个按钮：OK、Cancel、Apply Now以及Help（其中，在先前提出的演示中，Help按钮可能被隐藏）。当用户单击这些按钮时，页面的对话框进程将接收到通告消息。

属性页中的每个页面都是一个应用程序所定义的非模态对话框，这些页面可以管理用来查看与编辑项目属性的控件窗口。应该提供用来创建页面的对话框模板以及用来管理控件并设置相应项目属性的对话框进程。

当页面获取或失去焦点、用户单击OK、Cancel、Apply Now或Help按钮时，属性页将对对话框进程发送通告消息。这些通告消息是以WM_NOTIFY消息的形式进行发送的。lParam参数是NMHDR数据结构的地址，在这个数据结构中包含有属性页对话框的窗口句柄。

有些通告消息要求页面在响应WM_NOTIFY消息时返回TRUE或FALSE。在这种情况下，页面就必须使用SetWindowLong函数来设置页面对话框的DWL_MSGRESULT值为TRUE或FALSE。

21.1.2 页面

属性页必须至少包含一个页面，同时，其中所包含的页面数不得多于Win32头文件中所定义的MAXPROPPAGES值。属性页中的每个页面都有一个基于0的索引，这个索引值是属性页根据页面被添加到属性页中的顺序进行设置的。页面的索引值在向属性页发送的消息中被使用。

属性页面中可以包含嵌套的对话框。如果在属性页面中包含有嵌套对话框，那么必须为顶层的对话框设置WS_EX_CONTROLPARENT样式，并且用父对话框的句柄来调用IsDialogMessage函数。这样就能够确保用户使用记忆功能以及对对话框导航键来将焦点转移到嵌套对话框中的控件。

每个页面都有相应的图标与标号。属性页为每个页面创建一个标签（tab），并且在标签上显示图标与标号。所有的属性页页面都应该不使用黑色字体。为了确保所使用的字体不是黑色字体，应该在对话框模板中指定DS_3DLOOK样式。

页面的对话框进程不能调用EndDialog函数，否则，将会破坏整个属性页，而不仅仅是这个页面。

属性页页面最小尺寸是水平方向上212对话框单位以及在竖直方向上114对话框单位。如果某个页面对话框比这个尺寸更小，那么这个页面将被扩大到最小尺寸。在Prsht.h头文件中包含有关于属性页页面的三个推荐大小值。其中，PROP_SM_CXDLG与PROP_SM_CYDLG用来定义小型属性页页面的大小；PROP_MED_CXDLG与PROP_MED_CYDLG用来定义中型属性页页面的大小；PROP_LG_CXDLG与PROP_LG_CYDLG用来定义大型属性页页面的大小。使用这些推荐的大小设置，能够帮助确保自己的应用程序与其他Microsoft Windows应用程序页面大小的一致性。

可以使用下列值来设置属性页页面中元素的大小：

PROP_SM_CXDLG	小型属性页页面的宽度，以对话框为单位。
PROP_SM_CYDLG	小型属性页页面的高度，以对话框为单位。
PROP_MED_CXDLG	中型属性页页面的宽度，以对话框为单位。
PROP_MED_CYDLG	中型属性页页面的高度，以对话框为单位。
PROP_LG_CXDLG	大型属性页页面的宽度，以对话框为单位。
PROP_LG_CYDLG	大型属性页页面的高度，以对话框为单位。

21.1.3 创建属性页

在创建属性页之前，必须定义一个或多个页面。定义页面的过程包括利用关于本页面的信息（页面的图标、标号、对话框模板、对话框进程等等）来填充PROPSHEETPAGE数据结构，然后在调用CreatePropertySheetPage函数时指定数据结构的地址。这个函数将返回用来唯一地标识本页面的HPROPSHEETPAGE数据类型的句柄。

为了创建属性页，应该在对PropertySheet函数调用的过程中指定PROPSHEETHEADER数据结构地址。这个数据结构定义了属性页的图标与标题，并且在其中包含有HPROPSHEETPAGE句柄数组的地址。当PropertySheet创建属性页时，它将包含在上述数组中所定义的页面。页面在

属性页中出现的顺序与它们存储在数组中的顺序相同。

另一个创建属性页的方法是指定PROPSHEETPAGE数据结构数组，而不是指定HPROPSHEETPAGE句柄数组。在这种情况下，PropertySheet将在把页面添加到属性页中之前创建页面的句柄。

页面被创建之后，这个页面的对话框进程将接收到一个WM_INITDIALOG消息。这个消息的IParam参数是一个指向PROPSHEETPAGE数据结构副本的指针，这个数据结构在页面被创建时定义。特殊情况下，当一个页面被创建时，其数据结构的IParam数据成员可用来向对话框进程传递应用程序所定义的信息。除了IParam数据成员之外，这个数据结构应该当做只读数据结构看待。对除了IParam数据成员之外的其他数据成员进行的任何修改都可能会导致不可预知的后果。

在系统向应用程序传递页面PROPSHEETPAGE数据结构的副本时，它将使用相同的指针进行传递。这时，对数据结构所做的任何改变也将被一同传递。由于IParam数据成员被系统所忽略，因此这个数据成员可以非常安全地被修改，用来向应用程序的其他部分发送信息。例如，可以利用IParam数据成员向页面的信号回调函数传递信息。

PropertySheet能够自动地设置属性页的大小与初始位置。其中，初始位置的设置是根据所有者窗口的位置进行设置的，而大小则是根据属性页被创建时在页面数组中所指定的最大页面进行设置的。如果希望页面与属性页底部的四个按钮宽度相匹配，那么就应该将最宽页面的宽度设置为190对话框单位。

21.1.4 添加与删除页面

在创建属性页之后，应用程序就可以利用PSM_ADDPAGE消息来添加页面。需要说明的是，在属性页被创建之后，它的大小就不能改变；因此，新的页面不能大于当前在属性页中的最大页面。

应用程序可以使用PSM_REMOVEPAGE消息删除一个页面。在定义页面时，可以指定属性页在创建或删除这个页面时将要调用的PropSheetPageProc信号回调函数的地址。因此，PropSheetPageProc函数为进行单个页面的初始化操作以及删除操作提供了可能。

在析构一个属性页时，它将自动地析构被添加到这个属性页中的所有页面。此外，这些页面是按照创建这些页面的相反顺序进行析构的。如果需要析构一个已经被CreatePropertySheetPage函数所创建，但尚未添加到属性页中的页面，那么就应该使用DestroyPropertySheetPage函数。

21.1.5 属性页标题与页面标号

可以在用来创建属性页的PROPSHEETHEADER数据结构中指定属性页的标题。如果dwFlags数据成员包含有PSH_PROPTITLE值，那么属性页将为指定的标题字符串添加前缀“Properties for”。在属性页被创建之后，可以使用PSM_SETTITLE消息来改变属性页标题。

缺省情况下，属性页将使用对话框模板中所指定的名字字符串作为页面的标号。可以在用来定义这个页面的PROPSHEETPAGE数据结构dwFlags数据成员中包含PSP_USETITLE值。在指定

PSP_USETITLE之后, pszTitle数据成员就必须包含页面标题字符串的地址。

21.1.6 激活页面

在某一时刻, 属性页只能有一个活跃页面。处于活跃动态的那个页面将位于属性页所有页面重叠堆栈中的前端。用户可以通过选择页面的标签方法来激活某个页面, 而应用程序则使用 PSM_SETCURSEL消息来激活某个页面。

对于将要失去激活状态页面, 属性页将向这个页面发送PSN_KELLACTIVE通告消息。页面在对这个通告消息进行响应时, 应该确认用户对本页面所做的任何改变。如果页面需要在失去活跃状态之前请求用户进行更多的输入, 那么就应该使用SetWindowLong函数将页面的DWL_MSGRESULT值设置为TRUE。此外, 页面还应该显示一个消息框, 用来描述所存在的问题并且提供建议执行的动作。当页面可以失去活跃状态时, 应该将DWL_MSGRESULT的值设置为FALSE。

在获取活跃状态的页面可见之前, 属性页将向这个页面发送PSN_SETACTIVE通告消息。这个页面在对本通告消息进行响应时应该初始化自己的控件窗口。

21.1.7 帮助按钮

属性页可以显示两个帮助按钮。其中一个属性页帮助按钮显示在属性页框架的底部, 靠近OK/Cancel/Apply按钮; 另一个帮助按钮是标准的标题条按钮, 用来提供上下文敏感的帮助信息。

属性页的帮助按钮是可选的, 可以根据页面的实际需要来使用。应该按照以下的方法来在一个或多个页面中显示属性页帮助按钮:

- 在属性页PROPSHEETHEADER数据结构的dwFlags数据成员中设置PSH_HASHELP标记。
- 为每个将要显示帮助按钮的页面, 在页面PROPSHEETPAGE数据结构的dwFlags数据成员中设置PSP_HASHELP标记。

当用户单击Help按钮时, 活跃页面将接收到PSN_HELP通告消息。这个页面就应该显示Help帮助信息, 典型情况下是调用WinHelp函数。

删除标题条帮助按钮

缺省情况下, 标题条帮助按钮都会被显示出来, 因此对于OK/Cancel/Apply按钮来说, 上下文敏感的帮助信息总是有效的。但是, 这个按钮也可以被删除。如果需要删除属性页的标题条帮助按钮, 必须:

- 对于5.80版本之前的通用控件, 必须自己实现一个属性页信号回调函数。
- 对于5.80版本的通用控件及其后续通用控件, 只要设置属性页PROPSHEETHEADER数据结构dwFlags数据成员中的PSH_NOCONTEXTHELP标记即可。但是, 如果需要对5.80版本更早的通用控件提供兼容, 那么就应该自己实现信号回调函数。

如果需要实现一个用来删除标题条帮助按钮的属性页信号回调函数, 必须:

- 设置属性页PROPSHEETHEADER数据结构dwFlags数据成员中的PSH_USECALLBACK标记。

- 将PROPSHEETHEADER数据结构中的pfnCallback数据成员设置为指向信号回调函数。
- 实现信号回调函数。当这个函数接收到PSCB_PRECREATE消息时，它将获取一个指向属性页对话框模板的指针。应该从这个模板中删除DS_CONTEXTHELP样式。

以下例子演示了如何实现这样的信号回调函数：

```
int CALLBACK RemoveContextHelpProc(HWND hwnd,UINT message,LPARAM lParam)
{
    switch (message){
        case PSCB_PRECREATE:
            //Remove the DS_CONTEXTHELP style from the dialog template
            if (((LPDLGTEMPLATEEX)lParam)->signature ==0xFFFF){
                ((LPDLGTEMPLATEEX)lParam)->style &=~DS_CONTEXTHELP;
            }
            else {
                ((LPDLGTEMPLATE)lParam)->style &=~DS_CONTEXTHELP;
            }
            return TRUE;
        }
    return TRUE;
}
```

如果没有定义DLGTEMPLATEEX数据结构，那么就应该包含以下声明语句：

```
#include <pshpack1.h>
typedef struct DLGTEMPLATEEX
{
    WORD dlgVer;
    WORD signature;
    DWORD helpID;
    DWORD exStyle;
    DWORD style;
    WORD cDlgItems;
    short x;
    short y;
    short cx;
    short cy;
}DLGTEMPLATEEX,*LPDLGTEMPLATEEX;
#include <poppack.h>
```

21.1.8 确定、取消与立刻应用按钮

OK（确定）按钮与Apply Now（立刻应用）按钮相似，它们都用指示属性页面对用户所做的属性修改进行合法性检查并应用所做的修改。两者之间唯一的不同之处是对OK按钮的单击将会导致属性页在应用属性修改之后被析构。

当用户单击OK按钮或Apply Now按钮时，属性页将发送PSN_KILLACTIVE通告消息激活页面，从而能够使得页面可以对用户所做的属性改变进行合法性检查与确认。如果页面判断用户所做的改变合法，那么它将调用SetWindowLong函数将页面的DWL_MSGRESULT值设置为

FALSE。在这种情况下，属性页将向每个页面发送PSN_APPLY通告消息，要求这些页面将新属性应用于相应的项目。如果页面判断到用户所做的属性改变不合法，它将把DWL_MSGRESULT设置为TRUE，并且显示一个对话框用来告知用户所存在的问题。这时，页面将一直保持活跃状态，直到在对PSN_KILLACTIVE消息进行响应的过程中将DWL_MSGRESULT设置为FALSE。应用程序可以使用PSM_APPLY消息来模拟Apply Now按钮被选中的情形。

当页面开始变为活跃状态时，Apply Now按钮是关闭的，表示目前还没有任何属性改变信息可以应用。当页面接收到表示用户已经对某个属性进行修改的输入信号之后，页面就应该向属性页发送PSM_CHANGED消息。这个消息将导致属性页打开Apply Now按钮。然后，如果用户单击了Apply Now按钮或Cancel按钮，那么页面将重新初始化自己的控件并且再次发送PSM_UNCHANGED消息来关闭Apply Now按钮。

有时，Apply Now按钮将会导致一个页面改变属性页并且所做的改变不能恢复。在这种情况下，页面就应该向属性页发送PSM_CANCELTOCLOSE消息。这个消息将会使得属性页将OK按钮的文本改变为“Close”（关闭），表示所做的修改不能被取消。

有时，对页面所做的修改会影响到系统配置，从而需要在改变生效之前重新启动Windows或者重新启动系统。在做了这种改变之后，页面就应该向属性页发送PSM_RESTARTWINDOWS消息或PSM_REBOOTSYSTEM消息。这些消息会导致PropertySheet函数在属性页被析构之后返回ID_PSRESTARTWINDOWS或ID_PSREBOOTSYSTEM值。

当用户单击Cancel按钮时，属性页将向所有的页面发送PSN_RESET通告消息，用来说明属性页将要被析构。页面应该使用这个通告消息来执行清除操作。

21.1.9 属性页扩展

属性页扩展是一个能够向由另一个模块所创建的属性页中添加一个或多个页面的动态链接库（DLL）。创建属性页的模块中包含有一个扩展DLL调用来添加页面的AddPropSheetPageProc信号回调函数。这个函数能够接收页面的句柄并且接收一个由应用程序所定义的32位值。

扩展DLL中还包含有一个名为ExtensionPropSheetPageProc的信号回调函数，这个信号回调函数可以从创建属性页的模块中获取AddPropSheetPageProc的地址。此外，扩展DLL必须按名导出ExtensionPropSheetPageProc。

Windows头文件中包含有两个用来定义属性页信号回调函数的原型。如果需要定义AddPropSheetPageProc，就应该使用以下原型：

```
typedef BOOL (CALLBACK FAR * LPFNADDPSPSHEETPAGE)(HPROPSHEETPAGE, LPARAM);
```

如果需要定义ExtensionPropSheetPageProc，就应该使用以下原型：

```
typedef BOOL (CALLBACK FAR * LPFNADDPSPSHEETPAGES)(LPVOID, LPFNADDPSPSHEETPAGE, LPARAM);
```

21.2 使用属性页

本节包含用来演示如何创建属性页以及如何处理通告消息的例子。

21.2.1 创建属性页

本节中的这个例子将创建包含两个页面的属性页，其中一个页面用来设置电子数据表单元的字体系数，另一个页面用来设置电子数据表单元的边界属性。通过填充两个PROPSHEETPAGE数据结构以及指定将要被传递给PropertySheet函数的PROPSHEETHEADER数据结构，这个例子将定义页面。页面的对话框模板、图标以及标题是从包含在应用程序可执行文件中的资源进行加载的。属性页的图标也将从应用程序的资源中进行加载。

```
//DoPropertySheet -creates a property sheet that contains two pages.
//hwndOwner -handle to the owner window of the property sheet.
//
//Global variables
// g_hinst -instance handle
extern HINSTANCE g_hinst;

VOID DoPropertySheet(HWND hwndOwner)
{
    PROPSHEETPAGE psp [2 ];
    PROPSHEETHEADER psh;

    psp [0].dwSize =sizeof(PROPSHEETPAGE);
    psp [0].dwFlags =PSP_USEICONID |PSP_USETITLE;
    psp [0].hInstance =g_hinst;
    psp [0].pszTemplate =MAKEINTRESOURCE(DLG_FONT);
    psp [0].pszIcon =MAKEINTRESOURCE(IDI_FONT);
    psp [0].pfnDlgProc =FontDialogProc;
    psp [0].pszTitle =MAKEINTRESOURCE(IDS_FONT)
    psp [0].lParam =0;
    psp [0].pfnCallback =NULL;

    psp [1].dwSize =sizeof(PROPSHEETPAGE);
    psp [1].dwFlags =PSP_USEICONID |PSP_USETITLE;
    psp [1].hInstance =g_hinst;
    psp [1].pszTemplate =MAKEINTRESOURCE(DLG_BORDER);
    psp [1].pszIcon =MAKEINTRESOURCE(IDI_BORDER);
    psp [1].pfnDlgProc =BorderDialogProc;
    psp [1].pszTitle =MAKEINTRESOURCE(IDS_BORDER);
    psp [1].lParam =0;
    psp [1].pfnCallback =NULL;

    psh.dwSize =sizeof(PROPSHEETHEADER);
    psh.dwFlags =PSH_USEICONID |PSH_PROPSHEETPAGE;
    psh.hwndParent =hwndOwner;
    psh.hInstance =g_hinst;
    psh.pszIcon =MAKEINTRESOURCE(IDI_CELL_PROPERTIES);
```

```

psh.pszCaption =(LPSTR)'Cell Properties';
psh.nPages =sizeof(psp)/sizeof(PROPSHEETPAGE);
psh.nStartPage =0;
psh.psp =(LPCPROPSHEETPAGE)&psp;
psh.pfnCallback =NULL;

PropertySheet(&psh);
return;
}

```

21.2.2 处理通告消息

属性页通过发送WM_NOTIFY消息的方法来从页面获取信息,以及告知页面用户所做的动作。这个消息的lParam参数是NMHDR数据结构的地址,在这个数据结构中包含有属性对话框的句柄、页面对话框的句柄以及通告消息代码。页面必须将本页面的DWL_MSGRESULT值设置为TRUE或FALSE,从而对某些通告消息进行响应。

下面给出的代码是来自某个页面对话框处理程序的代码段,它演示了如何处理PSN_HELP通告消息:

```

case WM_NOTIFY:
    switch (((NMHDR FAR *)lParam)->code){

    case PSN_HELP:
        {
            char szBuf [FILE_LEN] ;//buffer for name of help file

            //Display help for the font properties page.
            LoadString(g_hinst,IDS_HELPFILE,&szBuf,FILE_LEN)
            WinHelp(((NMHDR FAR *)lParam)->hwndFrom,&szBuf,
                HELP_CONTEXT,IDH_FONT_PROPERTIES);
            break;
        }

        //Process other property sheet notifications here.
    }
}

```

21.3 在Internet Explorer中更新属性页

Microsoft Internet Explorer中的属性页支持以下新功能:

新通告消息

添加了PSN_GETOBJECT通告消息,从而允许某个页面是OLE拖动目标。

更新的数据结构

PROPSHEETHEADER数据结构与PROPSHEETPAGE数据结构已经被更新,从而能够支持

新的特性。关于这些数据结构的更多信息，请参见这些数据结构的参考。

21.4 属性页参考

21.4.1 属性页函数

AddPropSheetPageProc

这个参数用来指定由应用程序所定义的信号回调函数，所指定的信号回调函数将被属性页扩展用来向属性页中添加页面。

```
BOOL CALLBACK AddPropSheetPageProc (  
    HPROPSHEETPAGE hpage,  
    LPARAM lParam  
);
```

参数

hpage: 属性页页面的句柄。

lParam: 由应用程序所定义的32位值。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.1或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

导入库: 由用户定义。

CreatePropertySheetPage

为属性页创建一个新页面。

```
HPROPSHEETPAGE CreatePropertySheetPage (  
    LPCPROPSHEETPAGE lppsp  
);
```

参数

lppsp: PROPSHEETPAGE数据结构的地址，这个数据结构定义了将要包含在属性页中的页面。

返回值

如果操作成功，则返回给属性页页面的句柄；否则，返回NULL。

说明

应用程序可以使用PropertySheet函数来创建将要包含新属性页页面的属性页，或者使用PSM_ADDPAGE消息向一个现有的属性页中添加新页面。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

导入库: comctl32.lib。

DestroyPropertySheetPage

这个函数将析构一个属性页页面。应用程序必须为那些没有被传递给PropertySheet函数的页面调用这个函数。

```
BOOL DestroyPropertySheetPage (
    HPROPSHEETPAGE hPSPPage
);
```

参数

hPSPPage: 将要被析构的属性页页面句柄。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

导入库: comctl32.lib。

ExtensionPropSheetPageProc

这个函数指定将要接收AddPropSheetPageProc函数地址的由应用程序定义的信号回调函数, 所接收的函数位于创建这个属性页的模块中。属性页扩展必须按名称导出ExtensionPropSheetPageProc函数。

```
BOOL CALLBACK ExtensionPropSheetPageProc (
    LPVOID lpv,
    LPFNADDPROPSHEETPAGE lpfnAddPropSheetPageProc,
    LPARAM lParam
);
```

参数

lpv: 由应用程序所定义的值地址, 这个值用来描述将要为其创建属性页页面的项目。这个参数可以是NULL。

lpfnAddPropSheetPageProc: AddPropSheetPageProc函数的地址, 扩展DLL将调用这个函数来向属性页中添加页面。

lParam: 由应用程序所定义的32位值。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

说明

这个函数主要是为了实现向后兼容而提供的。属性页扩展处理程序应该使用AddPropSheetPageProc函数。

环境需求

Windows NT/2000: 需要使用Windows NT 3.1或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

导入库: 由用户定义。

PropertySheet

创建一个属性页, 并且在属性页中添加由属性页头结构中所定义的页面。

```
int PropertySheet (
    LPCPROPSHEETHEADER lppsh
);
```

参数

lppsh: 指向PROPSHEETHEADER数据结构的指针, 这个数据结构定义了属性页的框架与页面。

返回值

对于模态属性页, 如果操作成功, 则返回一个正值; 否则返回-1。

对于非模态属性页, 返回属性页窗口句柄。

下列返回值具有特殊的含义:

ID_PSREBOOTSYSTEM	表示一个页面向属性页发送了PSM_REBOOTSYSTEM消息。为了使得用户的改变发生作用, 必须重新启动计算机。
ID_PSRESTARTWINDOWS	表示一个页面向属性页发送了PSM_RESTARTWINDOWS消息。为了使得用户的改变发生作用, 必须重新启动Windows。

如果需要获取扩展的出错信息, 请调用GetLastError函数。

说明

缺省情况下, PropSheetPage函数创建模态对话框。如果PROPSHEETHEADER数据结构的dwFlags数据成员指定了PSH_MODELESS标志, 那么PropSheetPage将创建一个非模态对话框并且在创建这个对话框之后立刻返回。在这种情况下, PropSheetPage函数的返回值是非模态对话框的窗口句柄。

对于非模态属性页，消息循环应该使用PSM_ISDIALOGMESSAGE来将消息传递给属性页对话框。此外，消息循环还应该使用PSM_GETCURRENTPAGEHWND来决定何时析构对话框。当用户单击OK按钮或Cancel按钮时，PSM_GETCURRENTPAGEHWND将返回NULL。然后，就可以使用DestroyWindow函数来析构对话框。

在5.80版本中，PropSheetPage为模态属性页与非模态属性页携带不同的信息。在某些情况下，非模态属性页可能会需要在它们处于模态的情况下从PropSheetPage中所接收到的信息。特别地，非模态属性页需要知道自己应该返回ID_PSREBOOTSYSTEM还是ID_PSRESTARTWINDOWS。通过等待PSM_GETCURRENTPAGEHWND返回NULL，然后发送PSM_GETRESULT消息方法，非模态属性页可以接收到模态属性页能够从PropSheetPage中所接收到的数据值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

导入库：comctl32.lib。

PropSheetPageProc

指定在一个属性页页面被创建时以及在属性页页面将要被析构时，属性页所调用的由应用程序所定义的信号回调函数。应用程序可使用这个函数来执行页面的初始化操作与删除操作。

```
UINT CALLBACK PropSheetPageProc (
    HWND hwnd,
    UINT uMsg,
    LPPROPSHEETPAGE ppsp
);
```

参数

hwnd：这个参数是预留的参数，必须为NULL。

uMsg：[in] 动作标志，这个参数可以是下列值之一：

PSPCB_ADDREF 在5.80版本中有效，表示正在创建一个页面。返回值没有被使用。

PSPCB_CREATE 标志正在创建页面的对话框。如果允许创建页面对话框，则返回非零值；否则，返回0。

PSPCB_RELEASE 表示正在析构页面。返回值被忽略。

ppsp：[in/out] PROPSHEETPAGE数据结构的地址，这个数据结构定义了正在被创建或析构的页面。

返回值

返回值取决于uMsg参数的值。

说明

在调用CreatePropertySheetPage函数而指定PROPSHEETPAGE数据结构的地址之前，应用程序

序必须在这个数据结构的pfnCallback数据成员中指定这个信号回调函数的地址。

除了IParam数据成员之外，应用程序不应该对PROPSHEETPAGE数据结构中的任何其他部分进行修改。否则，将可能会导致不可预测的后果。IParam数据成员中包含有由应用程序所定义的数据，并且可以根据实际需要被修改。

环境需求

Windows NT/2000：需要使用Windows NT 3.1或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

导入库：由用户所定义。

PropSheetProc

当属性页正在被创建以及被初始化时，系统将调用这个由应用程序所定义的信号回调函数。

```
int CALLBACK PropSheetProc (
    HWND hwndDlg,
    UINT uMsg,
    LPARAM lParam
);
```

参数

hwndDlg：属性页对话框的句柄。

uMsg：正在被接收的消息，这个参数必须是下列值之一：

PSCB_INITIALIZED	这个值表示属性页正在被初始化，这个消息的IParam值为0。
PSCB_PRECREATE	这个值表示属性页将要被创建。这时，hwndDlg参数为NULL，IParam参数为内存中对话框模板的地址。这个模板是以DLGTEMPLATE数据结构的形式是存在的，在这个数据结构中有一个或多个DLGITEMTEMPLATE数据结构。

IParam：这个参数中包含有关于消息的更多信息，参数值的含义取决于uMsg参数。

返回值

返回值为0。

说明

为了打开PropSheetProc信号回调函数，应该在调用PropSheet函数时使用PROPSHEETHEADER数据结构创建属性页。利用其中的pfnCallback数据成员来指定信号回调函数的地址，并且在dwFlags数据成员中设置PSP_USECALLBACK标志。

PropSheetProc是应用程序定义函数名的占位符，PFNPROPSHEETCALLBACK数据类型是PropSheetProc信号回调函数的地址。

环境需求

Windows NT/2000：需要使用Windows NT 3.1或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

导入库: 由用户定义。

21.4.2 属性页消息

PSM_ADDPAGE

将一个新的页面添加到现有属性页中。可以显式地发送这个消息, 也可以使用PropSheet_AddPage宏来发送这个消息。

```
PSM_ADDPAGE
    wParam = 0;
    lParam (LPARAM) (HPROPSHEETPAGE) hpage;
```

参数

hpage: 将要添加页面的句柄。必须在先前已经调用CreatePropertySheetPage函数创建了这个页面。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

说明

由于属性页不能改变自己的大小来适应新的页面; 因此, 新页面大小必须不超过属性页中的当前最大页面。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_APPLY

模拟对Apply按钮的选择, 表示有一个或多个页面发生了改变, 并且这些改变需要进行合法性检查并记录下来。

```
PSM_APPLY
    wParam = 0;
    lParam = 0;
```

返回值

如果所有的页面都成功地应用了所做的改变, 则返回TRUE; 否则, 返回FALSE。

说明

属性页向当前页面发送PSN_KILLACTIVE通告消息。如果当前页面返回FALSE, 那么属性页将向所有的页面发送PSN_APPLY通告消息。可以显式地发送这个消息, 也可以使用

PropSheet_Apply宏来发送这个消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

PSM_CANCELTOCLOSE

如果自从上次执行PSN_APPLY通告消息以来已经进行了其他的改变, 并且这些改变不能被取消, 那么应用程序将发送这个通告消息。可以显式地发送这个消息, 也可以使用PropSheet_CancelToClose宏来发送这个消息。

PSM_CANCELTOCLOSE

wParam = 0;

lParam = 0;

返回值

没有返回值。

说明

PSM_CANCELTOCLOSE通告消息将关闭Cancel按钮, 并且将OK按钮的文本改变为“Close”。

大多数属性页都将会等到接收到PSN_APPLY通告消息之后才执行不可复原的改变操作。但是, 在某些情况下, 属性页面将会不按照标准的PSN_APPLY/PSN_RESET序列来进行不可复原的操作。一个例子就是包含有用来显示对话框窗口进行属性编辑的Edit按钮的属性页。当用户单击OK按钮来提交所做的改变时, 那么属性页页面将具有以下可选的动作:

- 它将记录所做的改变, 但直到接收到一个PSN_APPLY通告消息之后才应用这些改变。这是最受欢迎的处理方法。
- 在子对话框编辑完成之后, 立刻应用所做的改变, 但改变记住先前的设置。如果接收到一个PSN_RESET通告消息, 那么这些先前设置就可以用来恢复到先前状态。
- 立刻应用所做的改变, 并且在接收到PSN_RESET通告消息之后不试图恢复到先前设置, 这种情况有时对于前面两个操作不实际时有必要使用。

对于第三种选项, 应用程序应该向属性页发送一个PSM_CANCELTOCLOSE消息。这个消息用来告知用户对子对话框所做的改变在以后不能单击Cancel按钮进行恢复。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

PSM_CHANGED

这个通告消息用来告知属性页, 某个页面中的信息发生了改变。可以显式地发送这个消息,

也可以使用PropSheet_Changed宏来发送这个消息。

```
PSM_CHANGED
wParam = (WPARAM)(HWND) hwndPage;
lParam = 0;
```

参数

hwndPage: 发生改变的属性页页面句柄。

返回值

没有返回值。

说明

在这种情况下, 属性页将会打开Apply按钮。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

PSM_GETCURRENTPAGEHWND

获取属性页当前页面的窗口句柄。可以显式地发送这个消息, 也可以使用PropSheet_GetCurrentPageHwnd宏来发送这个消息。

```
PSM_GETCURRENTPAGEHWND
wParam = 0;
lParam = 0;
```

返回值

返回当前属性页页面的窗口句柄。

说明

对于非模态属性页, 应该使用PSM_GETCURRENTPAGEHWND消息来决定何时析构对话框。当用户单击OK按钮或Cancel按钮时, PSM_GETCURRENTPAGEHWND将返回NULL, 然后就可以使用DestroyWindow函数来析构这个对话框。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

参见

PropertySheet。

PSM_GETTABCONTROL

获取属性页标签控件的句柄。可以显式地发送这个消息, 也可以使用PropSheet_GetTabControl

宏来发送这个消息。

```
PSM_GETTABCONTROL
    wParam = 0;
    lParam = 0;
```

返回值

返回标签控件的句柄。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_HWNDTOINDEX

获取属性页页面的窗口句柄, 然后返回这个页面基于0的索引。可以显式地发送这个消息, 也可以使用PropSheet_HwndToIndex宏来发送这个消息。

```
PSM_HWNDTOINDEX
    wParam = (WPARAM)(HWND) hPageDlg;
```

参数

hPageDlg: 页面窗口的句柄。

返回值

如果操作成功, 则返回hPageDlg所指定属性页页面的基于0的索引; 否则, 返回-1。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

PSM_IDTOINDEX

获取属性页页面的资源ID, 并返回页面基于0的索引。可以显式地发送这个消息, 也可以使用PropSheet_IdToIndex宏来发送这个消息。

```
PSM_IDTOINDEX
    lParam = (LPARAM)(int) iPageID;
```

参数

iPageID: 页面的资源ID。

返回值

如果操作成功，则返回由iPageID参数所指定的属性页页面基于0的索引；否则，返回-1。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在prsht.h中进行了声明。

PSM_INDEXTOHWND

获取属性页页面的索引，并且返回这个页面的窗口句柄。可以显式地发送这个消息，也可以使用PropSheet_IndexToHwnd宏来发送这个消息。

```
PSM_INDEXTOHWND
wParam = (WPARAM)(int) iPageIndex;
```

参数

iPageIndex：页面基于0的索引。

返回值

如果操作成功，则返回由iPageIndex参数所指定的属性页页面的窗口句柄；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在prsht.h中进行了声明。

PSM_INDEXTOID

获取属性页页面的索引，并返回页面的资源ID。可以显式地发送这个消息，也可以使用PropSheet_IndexToId宏来发送这个消息。

```
PSM_INDEXTOID
wParam = (WPARAM)(int) iPageIndex;
```

参数

iPageIndex：页面基于0的索引。

返回值

如果操作成功，则返回由iPageIndex参数所指定的属性页面的资源ID；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在prsh.h中进行了声明。

PSM_INDEXTOPAGE

获取属性页页面的索引，并返回页面HPROPSHEETPAGE句柄。可以显式地发送这个消息，也可以使用PropSheet_IndexToPage宏来发送这个消息。

```
PSM_INDEXTOPAGE
    wParam = (WPARAM)(int) iPageIndex;
```

参数

iPageIndex：页面基于0的索引。

返回值

如果操作成功，则返回由iPageIndex参数所指定的属性页页面的HPROPSHEETPAGE句柄；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在prsh.h中进行了声明。

PSM_INSERTPAGE

向一个现有属性页中插入新页面，这个新页面可以插入到指定的索引位置或插入到指定页面的后面。可以显式地发送这个消息，也可以使用PropSheet_InsertPage宏来发送这个消息。

```
PSM_INSERTPAGE
    wParam = (WPARAM) index;
    - or -
    wParam = (WPARAM)(HPROPSHEETPAGE) hpageInsertAfter

    lParam = (LPARAM)(HPROPSHEETPAGE) hpage;
```

参数

wParam: 页面将要插入的位置, 如果将这个参数设置为NULL, 将使得新页面作为第一个页面。如果需要指定新页面将要被插入的位置, 可以向这个参数传递一个索引值或传递现有页面的HPROPSHEETPAGE句柄。

index: 如果wParam参数小于MAXUSHORT, 那么它将为新页面指定基于0的索引。例如, 如果需要使得被插入的页面位于属性页中的第三个位置, 那么就应该将index设置为2。如果需要将插入的页面成为第一个页面, 那么就应该将index设置为0。如果index拥有一个大于属性页中页面总数但小于MAXUSHORT的值, 那么这个页面将被放置在属性页的最后。

hpageInsertAfter: 如果将wParam参数设置为某个现有页面的HPROPSHEETPAGE句柄, 那么新页面将被插入在这个页面的后面。

hpage: 将要被插入的页面的句柄。当然, 首先必须调用CreatePropertySheetPage函数来创建这个页面。

返回值

如果页面被成功地插入, 则返回非零值; 否则, 返回0。

说明

位于插入点后面的那个页面将向右移动, 从而能够腾出空间放置新页面。

属性页并不会修改自己的大小来适合新页面。因此, 不应该使新页面大小大于属性页的最大页面。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PSM_ISDIALOGMESSAGE

属性页对话框中传递一个消息, 并且指示对话框是否处理这个消息。可以显式地发送这个消息, 也可以使用PropSheet_IsDialogMessage宏来发送这个消息。

```
PSM_ISDIALOGMESSAGE
wParam = 0;
lParam = (LPARAM) pMsg;
```

参数

pMsg: MSG数据结构的地址, 在这个数据结构中包含有将要被检查的消息。

返回值

如果这个消息被成功地处理, 则返回TRUE; 否则, 返回FALSE。

说明

为了使得非模态属性页能够向属性页对话框传递消息，消息循环应该使用PSM_ISDIALOGMESSAGE消息。

如果返回值表明消息已经被处理，那么这个消息就不应该传递给TranslateMessage函数或DispatchMessage函数。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

参见

PropertySheet。

PSM_PAGETOINDEX

获取属性页页面的HPROPSHEETPAGE句柄，并返回这个页面基于0的索引。可以显式地发送这个消息，也可以使用PropSheet_PageToIndex宏来发送这个消息。

```
PSM_PAGETOINDEX
lParam = (LPARAM)(HPROPSHEETPAGE) hPage;
```

参数

hPage：属性页页面的HPROPSHEETPAGE句柄。

返回值

如果操作成功，就返回由hPage参数所指定的属性页页面基于0的索引；否则，返回-1。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在prsh.h中进行了声明。

PSM_PRESSBUTTON

模拟属性页按钮的选择。可以显式地发送这个消息，也可以使用PropSheet_PressButton宏来发送这个消息。

```
PSM_PRESSBUTTON
wParam = (WPARAM)(int) iButton;
lParam = 0;
```


参数

iButton: 将要被选择按钮的索引, 这个参数必须是下列值之一:

PSBTN_APPLYNOW	表示选择了Apply按钮。
PSBTN_BACK	表示选择了Back按钮。
PSBTN_CANCEL	表示选择了Cancel按钮。
PSBTN_FINISH	表示选择了Finish按钮。
PSBTN_HELP	表示选择了Help按钮。
PSBTN_NEXT	表示选择了Next按钮。
PSBTN_OK	表示选择了OK按钮。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_QUERYSIBLINGS

这个消息将被发送给属性页, 然后属性页将把这个消息转发给其中的每个页面。可以显式地发送这个消息, 也可以使用PropSheet_QuerySiblings宏来发送这个消息。

```
PSM_QUERYSIBLINGS
    wParam = (WPARAM) param1;
    lParam = (LPARAM) param2;
```

参数

param1: 应用程序定义的第一个参数。

param2: 应用程序定义的第二个参数。

返回值

返回来自属性页中某个页面的非零值, 如果没有属性页页面返回非零值, 则返回0。

说明

如果某个页面返回那一个非零值, 那么属性页将不向后续页面发送这个消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_REBOOTSYSTEM

这个消息用来说明为了使得所做的修改生效, 必须重新启动系统。可以显式地发送PSM_

REBOOTSYSTEM消息，也可以使用PropSheet_RebootSystem宏来发送这个消息。

```
PSM_REBOOTSYSTEM
    wParam = 0;
    lParam = 0;
```

返回值

没有返回值。

说明

应用程序只有在对PSN_APPLY通告消息或PSN_KILLACTIVE通告消息进行响应时发送这个消息。

这个消息将会导致PropSheet函数返回ID_PSREBOOTSYSTEM值，但只有在用户单击OK按钮关闭属性页时才会发生这个动作。对系统进行重新启动是由应用程序来完成的，可以使用ExitWindowEx函数来完成这个动作。

这个消息用来取代消息之前与之后的所有PSM_RESTARTWINDOWS消息。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

PSM_REMOVEPAGE

从属性页中删除一个页面。可以显式地发送这个消息，也可以使用PropSheet_RemovePage宏来发送这个消息。

```
PSM_REMOVEPAGE
    wParam = (WPARAM)(int) index;
    lParam = (LPARAM)(HPROPSHEETPAGE) hpage;
```

参数

index和hpage：这两个参数分别是要将被删除页面的基于0的索引以及这个页面的句柄。一个应用程序可以指定页面的索引或指定页面的句柄，也可以两者都指定。如果这两者都指定了，那么优先考虑hpage参数。

返回值

没有返回值。

说明

发送PSM_REMOVEPAGE消息将会析构正在被删除的属性页页面。

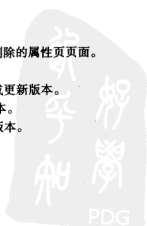
环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。



PSM_RESTARTWINDOWS

这个消息用来表明为了使得所做的改变生效，必须重新启动Windows。

PSM_RESTARTWINDOWS

wParam = 0;

lParam = 0;

返回值

没有返回值。

说明

只有在响应PSN_APPLY通告消息或PSN_KILLACTIVE通告消息时，应用程序才应该发送这个消息。可以显式地发送PSM_RESTARTWINDOWS消息，也可以使用PropSheet_RestartWindows宏来发送这个消息。

这个消息将会导致PropertySheet函数返回ID_PSRESTARTWINDOWS值，但只有在用户单击OK按钮关闭属性页之后，这个动作才会发生。应用程序负责重新启动Windows，它将调用ExitWindowEx函数来完成这个重新启动的动作。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

PSM_SETCURSEL

激活属性页中指定的页面。可以显式地发送这个消息，也可以使用PropSheet_SetCurSel宏来发送这个消息。

PSM_SETCURSEL

wParam = (WPARAM)(int) index;

lParam = (LPARAM)(HPROPSHEETPAGE) hpage;

参数

index和hpage：这两个参数分别表示将要被激活页面的基于0的索引以及页面的句柄。应用程序可以指定索引值或指定句柄，也可以两者都指定。如果两者都指定了，那么将优先考虑hpage参数。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

将要失去激活状态的那个窗口将接收到PSN_KILLACTIVE通告消息，而将要获取到激活状态的那个窗口将接收到PSN_SETACTIVE通告消息。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_SETCURSELID

根据页面的资源ID来激活属性页中的给定页面。可以显式地发送这个消息, 也可以使用PropSheet_SetCurSelByID宏来发送这个消息。

```
PSM_SETCURSELID  
wParam = 0;  
lParam = (LPARAM)(int) id;
```

参数

id: 将要被激活页面的资源ID。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

说明

将要失去激活状态的那个窗口将接收到PSN_KILLACTIVE通告消息, 而将要获取到激活状态的那个窗口将接收到PSN_SETACTIVE通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_SETFINISHTEXT

设置向导中Finish按钮的文本, 显示并打开这个按钮, 隐藏Next按钮与Back按钮。可以显式地发送这个消息, 也可以使用PropSheet_SetFinishText宏来发送这个消息。

```
PSM_SETFINISHTEXT  
wParam = 0;  
lParam = (LPARAM)(LPSTR) lpszText;
```

参数

lpszText: Finish按钮新文本的地址。

返回值

没有返回值。

说明

这个消息将导致DM_SETDEFID消息被发送给向导的对话框。其中wParam参数用来指定Finish按钮的标识符。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSM_SETHEADERSUBTITLE

设置向导内部页面头标的子标题文本。可以显式地发送这个消息, 也可以使用PropSheet_SetHeaderSubTitle宏来发送这个消息。

```
PSM_SETHEADERSUBTITLE
wParam = (WPARAM)(int) iPageIndex;
lParam = (LPARAM)(LPCTSTR) pszHeaderSubTitle;
```

参数

iPageIndex: 向导页面基于0的索引。

pszHeaderSubTitle: 新头标的子标题。

返回值

没有返回值。

说明

如果指定了当前页面, 这个页面将立刻进行重绘, 从而显示新的子标题。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PSM_HWNDTOINDEX、PSM_IDTOINDEX、PSM_PAGETOINDEX。

PSM_SETHEADERTITLE

设置向导内部页面头标的标题文本。可以显式地发送这个消息, 也可以使用PropSheet_SetHeaderTitle宏来发送这个消息。

```
PSM_SETHEADERTITLE
wParam = (WPARAM)(int) iPageIndex;
lParam = (LPARAM)(LPCTSTR) pszHeaderTitle;
```

参数

iPageIndex: 向导页面基于0的索引。

pszHeaderTitle: 新头标的子标题。

返回值

没有返回值。

说明

如果指定了当前页面，这个页面将立刻进行重绘，从而显示新的标题。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在prsh.h中进行了声明。

参见

PSM_HWNDTOINDEX、PSM_IDTOINDEX、PSM_PAGETOINDEX。

PSM_SETTITLE

设置属性页的标题。可以显式地发送这个消息，也可以使用PropSheet_SetTitle宏来发送这个消息。

PSM_SETTITLE

```
wParam = (WPARAM)(DWORD) dwStyle;
lParam = (LPARAM)(LPCSTR) lpszText;
```

参数

dwStyle：用来说明是否需要为指定的标题字符串添加前缀“Properties for”的标志。如果dwStyle参数为PSH_PROPTITLE值，那么将添加这个前缀。否则，将不添加这个前缀。

lpszText：包含标题字符串的缓冲器地址。如果这个参数的高位字部分为NULL，那么属性页将加载由低位字部分所指定的字符串资源。

返回值

没有返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

PSM_SETWIZBUTTONS

打开或关闭向导中的Back按钮、Next按钮与Finish按钮。也可以使用PropSheet_SetWizButtons宏来发送这些消息。

```

PSM_SETWIZBUTTONS
    wParam = 0;
    lParam = (LPARAM)(DWORD) dwFlags;

```

参数

dwFlags: 用来指明哪个属性页按钮被打开的标志值, 可以使用下列一个或多个值的组合:

PSWIZB_BACK 这个值表示打开Back按钮。如果没有设置这个标志位, 那么Back按钮将显示为关闭状态。

PSWIZB_DISABLEDFINISH 这个值表示关闭Finish按钮。

PSWIZB_FINISH 这个值表示打开Finish按钮。

PSWIZB_NEXT 这个值表示打开Next按钮。如果没有设置这个标志位, 那么Next按钮将显示为关闭状态。

返回值

没有返回值。

说明

如果通告消息处理程序使用PostMessage来发送PSM_SETWIZBUTTONS消息, 那么在处理程序返回之前就不应该执行任何会影响窗口焦点的动作。例如, 如果在使用PostMessage发送PSM_SETWIZBUTTONS消息之后立刻调用MessageBox函数, 那么消息框将接收到焦点。由于直到消息到达消息队列的头部时, 所发送的消息才会被传递, 因此要直到向导失去了消息框焦点之后才会传递PSM_SETWIZBUTTONS消息。所以, 属性页将有可能不能正确地设置按钮的焦点。

如果在处理PSN_SETACTIVE通告消息的过程中发送了PSM_SETWIZBUTTONS消息, 那么就应该使用PostMessage函数而不是使用SendMessage函数。否则, 系统将不能正确地更新按钮。如果使用PropSheet_SetWizButtons宏发送这个消息, 那么这个消息将被继续发送。在任何其他时刻, 可以使用SendMessage来发送PSM_SETWIZBUTTONS消息。

向导将在每个页面的底部显示三个或四个按钮。这个消息可用来指定哪个按钮将要被打开。通常, 向导将显示Back按钮、Cancel按钮以及一个Next按钮或Finish按钮。典型情况下, 仅仅为欢迎页面打开Next按钮, 为内部页面打开Next按钮与Back按钮, 为完成页面打开Back按钮与Finish按钮。而Cancel按钮总是处于打开状态。通常, 可以通过设置PSWIZB_FINISH或PSWIZB_DISABLEDFINISH的方法来用Finish按钮取代Next按钮。如果需要同时显示Next按钮与Finish按钮, 那么就应该在创建向导时给向导的PROPSHEETHEADER数据结构dwFlags数据成员设置PSH_WIZARDHASFINISHFLAG标志值。然后, 每个页面将显示所有的四个按钮。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在prsht.h中进行了声明。

PSM_UNCHANGED

这个消息用来告知属性页，某个页面中的信息已经恢复到先前保存的状态。可以显式地发送这个消息，也可以使用PropSheet_UnChanged宏来发送这个消息。

```
PSM_UNCHANGED
wParam = (WPARAM)(HWND) hwndPage;
lParam = 0;
```

参数

hwndPage: 已经恢复到先前保存状态的页面句柄。

返回值

没有返回值。

说明

如果没有任何其他页面向属性页注册所改变的信息，属性页将关闭Apply按钮。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

21.4.3 属性页宏

PropSheet_AddPage

将一个新页面添加到现有属性页的尾部。可以使用这个宏，也可以通过显式发送PSM_ADDPAGE消息的方法来完成此动作。

```
BOOL PropSheet_AddPage (
    HWND hPropSheetDlg,
    HPROPSHEETPAGE hpage,
);
```

参数

hPropSheetDlg: 属性页的句柄。

hpage: 将要被添加的页面的句柄。当然，这个页面首先必须被CreatePropertySheetPage函数所创建。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

由于属性页不会改变大小以适合新页面，因此新页面应该不大于属性页中的当前最大页面。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PropSheet_Apply

模拟Apply按钮被选中的动作, 用来表示一个或多个页面已经发生了改变, 并且这些改变需要进行合法性检查并记录下来。可以使用这个宏, 也可以通过显式发送PSM_APPLY消息的方法来完成此动作。

```
BOOL PropSheet_Apply (
    HWND hPropSheetDlg
);
```

参数

hPropSheetDlg: 属性页的句柄。

返回值

如果所有的页面都成功地应用了这些改变, 则返回TRUE; 否则, 返回FALSE。

说明

属性页将向当前页面发送PSN_KILLACTIVE通告消息。如果当前页面返回FALSE, 那么属性页将向所有的页面发送PSN_APPLY通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PropSheet_CancelToClose

如果自从最近一次PSN_APPLY通告消息发送以来发生了不可取消的改变, 则使用这个宏。也可以显式发送PSM_CANCELTOCLOSE消息来完成这个动作。

```
VOID PropSheet_CancelToClose (
    HWND hpropSheetDlg
);
```

参数

hPropSheetDlg: 属性页的句柄。

返回值

没有返回值。

说明

PSM_CANCELTOCLOSE将关闭Cancel按钮, 并且将OK按钮的文本改变为“Close”。可以使用这个宏, 或者通过显式地发送PSM_CANCELTOCLOSE消息的方法来完成这个动作。

大多数属性页直到接收PSN_APPLY通告消息之后才执行不可恢复的改变。但是, 在某些情

况下,属性页将会不按照标准的PSN_APPLY/PSN_RESET顺序执行不可恢复的改变。一个例子就是包含有用来显示子对话框窗口进行属性编辑的Edit按钮的属性页。当用户单击OK按钮来提交所做的改变时,那么属性页页面将具有以下可选的动作:

- 它将记录所做的改变,但直到接收到一个PSN_APPLY通告消息之后才应用这些改变。这是最受欢迎的处理方法。
- 在子对话框编辑完成之后,立刻应用所做的改变,但在改变之前必须记住先前的设置。如果接收到一个PSN_RESET通告消息,那么这些先前设置就可以用来恢复到先前状态。
- 立刻应用所做的改变,并且在接收到PSN_RESET通告消息之后不试图恢复到先前设置。这种情况有时对于前面两个操作不实际时有必要使用。

对于第三种选项,应用程序应该向属性页发送一个PSM_CANCELTOCLOSE消息。这个消息用来告知用户,对子对话框所做的改变在以后不能单击Cancel按钮进行恢复。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PropSheet_Changed

告知属性页,页面中的信息已经发生了改变。可以使用这个宏,也可以通过显式发送PSM_CHANGED消息的方法来完成此动作。

```
BOOL PropSheet_Changed (
    HWND hPropSheetDlg,
    HWND hwndPage,
);
```

参数

hPropSheetDlg: 属性页的句柄。

hwndPage: 已经发生改变的页面的句柄。

返回值

没有返回值。

说明

属性页将打开Apply按钮。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PropSheet_GetCurrentPageHwnd

获取属性页当前页面的窗口句柄。可以使用这个宏,也可以通过显式发送PSM_GETCURR-

ENTPAGEHWND消息的方法来完成此动作。

```
HWND PropSheet_GetCurrentPageHwnd (
    HWND hDlg
);
```

参数

hDlg: 属性页的句柄。

返回值

返回当前属性页窗口的句柄。

说明

对于非模态属性页, 可以使用PropSheet_GetCurrentPageHwnd宏来决定何时应该析构对话框。当用户单击OK按钮或Cancel按钮时, PropSheet_GetCurrentPageHwnd将返回NULL, 然后就可以使用DestroyWindow函数来析构这个对话框。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

参见

PropertySheet。

PropSheet_GetTabControl

获取属性页标签控件的句柄。可以使用这个宏, 也可以通过显式发送PSM_GETTABCONTROL消息的方法来完成此动作。

```
HWND PropSheet_GetTabControl (
    HWND hPropSheetDlg
);
```

参数

hPropSheetDlg: 属性页的句柄。

返回值

返回标签控件的句柄。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

PropSheet_HwndToIndex

获取属性页窗口的句柄, 然后返回这个页面基于0的索引。可以使用这个宏, 也可以通

过显式发送PSM_HWNDTOINDEX消息的方法来完成此动作。

```
int PropSheet_HwndToIndex (
    HWND hPropSheetDlg,
    HWND hPageDlg
);
```

参数

hPropSheetDlg: 属性页窗口的句柄。

hPageDlg: 页面窗口的句柄。

返回值

如果操作成功, 则返回由hPageDlg参数所指定的属性页页面基于0的索引; 否则, 返回-1。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsht.h中进行了声明。

参见

PropSheet_GetCurrentPageHwnd、GetParent。

PropSheet_IdToIndex

获取属性页页面的资源ID, 应返回这个页面基于0的索引。可以使用这个宏, 也可以通过显式发送PSM_IDTOINDEX消息的方法来完成此动作。

```
int PropSheet_IdToIndex (
    HWND hPropSheetDlg,
    int iPageID
);
```

参数

hPropSheetDlg: 属性页窗口的句柄。

iPageID: 页面的资源ID。

返回值

如果操作成功, 则返回由iPageID参数所指定的属性页页面基于0的索引; 否则, 返回-1。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的

Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PropSheet_IndexToId。

PropSheet_IndexToHwnd

获取属性页页面的索引, 然后返回页面的窗口句柄。可以使用这个宏, 也可以通过显式发送PSM_INDEXTOWND消息的方法来完成此动作。

```
HWND PropSheet_IndexToHwnd (
    HWND hPropSheetDlg,
    int iPageIndex
);
```

参数

hPropSheetDlg: 属性页页面窗口的句柄。

iPageIndex: 这个页面基于0的索引。

返回值

如果操作成功, 则返回由iPageIndex参数所指定属性页页面的句柄; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PropSheet_HwndToIndex。

PropSheet_IndexToId

获取属性页页面的索引, 然后返回这个页面的资源ID。可以使用这个宏, 也可以通过显式发送PSM_INDEXTOID消息的方法来完成此动作。

```
int PropSheet_IndexToId (
    HWND hPropSheetDlg,
    int iPageIndex
);
```

参数

hPropSheetDlg: 属性页句柄。

iPageIndex: 页面基于0的索引。

返回值

如果操作成功, 则返回由iPageIndex参数所指定属性页页面的资源ID; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PSM_INDEXTOID。

PropSheet_IndexToPage

获取属性页页面的索引, 然后返回这个页面的HPROPSHEETPAGE句柄。可以使用这个宏, 也可以通过显式发送PSM_INDEXTOPAGE消息的方法来完成此动作。

```
HPROPSHEETPAGE PropSheet_IndexToPage (  
    HWND hPropSheetDlg,  
    int iPageIndex  
);
```

参数

hPropSheetDlg: 属性页窗口的句柄。

iPageIndex: 页面基于0的索引。

返回值

如果操作成功, 返回由iPageIndex参数所指定属性页页面的HPROPSHEETPAGE句柄; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PropSheet_PageToIndex。

PropSheet_InsertPage

向一个现有属性页中插入新页面。这个新页面可以插入在一个指定的索引号位置，也可以插入在指定页面的后面。可以使用这个宏，也可以通过显式发送PSM_INSERTPAGE消息的方法来完成此动作。

```

BOOL PropSheet_InsertPage (
    HWND hPropSheetDlg ,
    int index ,
    HPROPSHEETPAGE hpage
);

-or-

BOOL PropSheet_InsertPage (
    HWND hPropSheetDlg ,
    HPROPSHEETPAGE hpageInsertAfter ,
    HPROPSHEETPAGE hpage
);

```

参数

hPropSheetDlg: 属性页的句柄。

wParam: 页面将要被插入的位置。如果wParam参数被设置为NULL，则表示新页面将作为第一个页面。如果需要指定新页面被插入的位置，就可以传递一个索引值或现有页面的HPROPSHEETPAGE句柄。

index: 如果wParam参数小于MAXUSHORT，这个参数将用来指定新页面基于0的索引。例如，如果需要使得被插入的页面位于属性页的第三个位置，就应该将index参数设置为2。如果需要使新页面成为属性页的第一个页面，则应该将index参数设置为0。如果index的值大于属性页中的页面总数，但小于MAXUSHORT，那么新页面将被添加到属性页最后。

hpageInsertAfter: 如果将wParam参数设置为一个现有页面的HPROPSHEETPAGE句柄，那么新页面将插入到这个页面的后面。

hpage: 将要被插入的页面句柄。首先必须调用CreatePropertySheetPage函数来创建这个页面。

返回值

如果页面被成功地插入，将返回一个非零值；否则，返回0。

说明

位于插入点后面的页面将向右移动，从而为新页面腾出空间。

由于属性页不会改变自己的大小来适合新页面，因此不应该使新页面大小大于属性页中的最大页面。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

PropSheet_IsDialogMessage

向属性页对话框传递一个消息, 并且说明对话框是否处理了这个消息。可以使用这个宏, 也可以通过显式发送PSM_ISDIALOGMESSAGE消息的方法来完成此动作。

```
BOOL PropSheet_IsDialogMessage (  
    HWND hDlg,  
    LPMSG pMsg  
);
```

参数

hDlg: 属性页的句柄。

pMsg: MSG数据结构的地址, 在这个数据结构中包含有将要被检查的消息。

返回值

如果这个宏已经被处理, 则返回TRUE; 否则, 如果这个宏没有被处理, 则返回FALSE。

说明

对于非模态属性页, 消息循环应该使用PSM_ISDIALOGMESSAGE消息来向属性页对话框传递消息。

如果返回值表明PSM_ISDIALOGMESSAGE消息没有被处理, 那么这个消息就不能被传递给TranslateMessage函数或DispatchMessage函数。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

参见

PropertySheet。

PropSheet_PageToIndex

获取属性页页面的HPROPSHEETPAGE句柄, 并且返回这个页面基于0的索引。可以使用这个宏, 也可以通过发送PSM_PAGETOINDEX消息的方法来完成此动作。

```
int PropSheet_PageToIndex (  
    HWND hPropSheetDlg,  
    HPROPSHEETPAGE hPage,  
);
```

参数

hPropSheetDlg: 属性页的句柄。

hPage: 属性页页面的HPROPSHEETPAGE句柄。

返回值

如果操作成功, 则返回由hPage所指定的属性页页面基于0的索引; 否则, 返回-1。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

CreatePropertySheetPage、PropSheet_IndexToPage。

PropSheet_PressButton

模拟属性页按钮的选择过程。可以使用这个宏, 也可以通过显式发送PSM_PRESSBUTTON消息的方法来完成此动作。

```
BOOL PropSheet_PressButton (
    HWND hPropSheetDlg,
    int iButton
);
```

参数

hPropSheetDlg: 属性页的句柄。

iButton: 将要被选择的按钮的索引, 这个参数可以是下列值之一:

PSBTN_APPLYNOW	这个值表示选择Apply按钮。
PSBTN_BACK	这个值表示选择Back按钮。
PSBTN_CANCEL	这个值表示选择Cancel按钮。
PSBTN_FINISH	这个值表示选择Finish按钮。
PSBTN_HELP	这个值表示选择Help按钮。
PSBTN_NEXT	这个值表示选择Next按钮。
PSBTN_OK	这个值表示选择OK按钮。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

PropSheet_QuerySiblings

这个宏将导致属性页向每个页面发送PSM_QUERYSIBLINGS消息。可以使用这个宏，也可以通过显式发送PSM_QUERYSIBLINGS消息的方法来完成此动作。

```
int PropSheet_QuerySiblings (
    HWND hPropSheetDlg,
    WPARAM param1,
    LPARAM param2
);
```

参数

hPropSheetDlg：属性页的句柄。

param1：应用程序所定义的第一个参数。

param2：用程序所定义的第二个参数。

返回值

返回属性页中某个页面的非零值；或者，如果没有页面能够返回非零值，则返回0。

说明

如果有某个页面返回一个非零值，那么这个属性页将不向后续的页面发送这个消息。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

PropSheet_RebootSystem

用来表示为了使得所做的改变生效，必须重新启动系统。可以使用这个宏，也可以通过显式发送PSM_REBOOTSYSTEM消息的方法来完成此动作。

```
VOID PropSheet_RebootSystem (
    HWND hPropSheetDlg,
);
```

参数

hPropSheetDlg：属性页的句柄。

返回值

没有返回值。

说明

只有在对PSN_APPLY通告消息或PSN_KILLACTIVE通告消息进行响应时，应用程序才能发送PSM_REBOOTSYSTEM消息。

这个宏将会导致PropertySheet函数返回ID_PSREBOOTSYSTEM值，但只有当用户单击OK

按钮用来关闭属性页之后，才能完成这个动作。应用程序负责系统的重新启动，它将调用ExitWindowsEx函数来完成这个动作。

这个宏用来取代在其之前或之后的所有PropSheet_RebootSystem宏。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

参见

PSM_RESTARTWINDOWS。

PropSheet_RemovePage

从属性页中删除一个页面。可以使用这个宏，也可以通过发送PSM_REMOVEPAGE消息的方法来完成此动作。

```
VOID PropSheet_RemovePage (
    HWND hPropSheetDlg,
    int index,
    HPROPSHEETPAGE hpage
);
```

参数

hPropSheetDlg：属性页的句柄。

index和hpage：这两个参数分别用来表示将要被删除的页面基于0的索引以及这个页面的句柄。应用程序可以指定页面的索引或句柄，也可以同时指定这两个参数。如果同时指定了这两个参数，那么hpage函数将优先处理。

返回值

没有返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

PropSheet_RestartWindows

发送一个PSM_RESTARTWINDOWS消息，用来表示为了使得所做的改变生效，必须重新启动Windows。可以使用这个宏，也可以通过发送PSM_RESTARTWINDOWS消息的方法来完成此动作。

```
VOID PropSheet_RestartWindows (
    HWND hPropSheetDlg
);
```

参数

hPropSheetDlg: 属性页的句柄。

返回值

没有返回值。

说明

只有在对PSN_APPLY通告消息或PSN_KILLACTIVE通告消息进行响应时, 应用程序才应该发送PSM_RESTARTWINDOWS消息。

PSM_RESTARTWINDOWS消息将会导致PropertySheet函数返回ID_PSRESTARTWINDOWS值, 但只有在用户单击OK按钮关闭属性页之后这个动作才会生效。应用程序负责重新启动Windows, 它将通过调用ExitWindowsEx函数的方法来完成这个动作。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

PropSheet_SetCurSel

激活属性页中的指定页面。可以使用这个宏, 也可以通过显式发送PSM_SETCURSEL消息的方法来完成此动作。

```
BOOL PropSheet_SetCurSel (  
    HWND hPropSheetDlg,  
    HPROPSHEETPAGE hpage,  
    int index  
);
```

参数

hPropSheetDlg: 属性页的句柄。

index和hpage: 这两个参数分别用来表示将要被激活的页面基于0的索引以及页面的句柄。一个应用程序可以指定页面的索引或这个页面的句柄, 也可以两个都指定。如果两个参数都指定了, 那么hpage参数将优先进行处理。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

说明

将要失去活跃状态的那个窗口将接收到PSN_KILLACTIVE通告消息, 而将要获取活跃状态的那个窗口将接收到PSN_SETACTIVE通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

PropSheet_SetCurSelByID

根据页面的资源ID，激活属性页中的指定页面。可以使用这个宏，也可以通过显式发送PSM_SETCURSELID消息的方法来完成此动作。

```
BOOL PropSheet_SetCurSelByID (
    HWND hPropSheetDlg,
    int id
);
```

参数

hPropSheetDlg：属性页的句柄。

id：将要被激活页面的资源ID。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

将要失去活跃状态的那个窗口将接收到PSN_KILLACTIVE通告消息，而将要获取活跃状态的那个窗口将接收到PSN_SETACTIVE通告消息。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsht.h中进行了声明。

PropSheet_SetFinishText

设置向导中Finish按钮的文本，显示并打开这个按钮，隐藏Next按钮与Back按钮。可以使用这个宏，也可以通过显式发送PSM_SETFINISHTEXT消息的方法来完成此动作。

```
VOID PropSheet_SetFinishText (
    HWND hPropSheetDlg,
    LPCTSTR lpszText
);
```

参数

hPropSheetDlg：向导的窗口句柄。

lpszText：Finish按钮的新文本地址。

返回值

没有返回值。

说明

这个宏将导致DM_SETDEFID消息被发送给属性页对话框。其中，wParam参数用来指定Finish按钮的ID。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PropSheet_SetHeaderSubTitle

设置向导内部页面的头标子标题文本。可以使用这个宏, 也可以通过显式发送PSM_SETHEADERSUBTITLE消息的方法来完成此动作。

```
VOID PropSheet_SetHeaderSubTitle (  
    HWND hWizardDlg,  
    int iPageIndex,  
    LPCSTR pszHeaderSubTitle  
);
```

参数

hWizardDlg: 向导窗口的句柄。

iPageIndex: 页面基于0的索引。

pszHeaderSubTitle: 新的头标子标题。

返回值

没有返回值。

说明

如果指定了当前页面, 它将立刻进行重绘并显示新的子标题。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PropSheet_HwndToIndex、PropSheet_IdToIndex、PropSheet_PageToIndex。

PropSheet_SetHeaderTitle

为向导的内部页面设置头标的标题文本。可以使用这个宏, 也可以通过显式发送PSM_SETHEADERTITLE消息的方法来完成此动作。

```
int PropSheet_SetHeaderTitle (  
    HWND hWizardDlg,
```

```
int iPageIndex,
LPCSTR pszHeaderTitle
);
```

参数

hWizardDlg: 向导窗口的句柄。

iPageIndex: 页面基于0的索引。

pszHeaderTitle: 新的头标标题。

返回值

没有返回值。

说明

如果指定了当前页面，它将立刻进行重绘并显示新的标题。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在prsht.h中进行了声明。

PropSheet_SetTitle

设置属性页的标题。可以使用这个宏, 也可以通过显式发送PSM_SETTITLE消息的方法来完成此动作。

```
VOID PropSheet_SetTitle (
    HWND hPropSheetDlg,
    DWORD dwStyle,
    LPCSTR lpszText
);
```

参数

hPropSheetDlg: 属性页的句柄。

dwStyle: 用来说明是否需要在指定的标题字符串中添加前缀“Properties for”的标志。如果dwStyle的值为PSH_PROPTITLE, 那么就需要包括这个前缀。否则, 就不需要包括这个前缀。

lpszText: 包含有标题字符串的缓冲器地址。如果这个参数的高位字部分为NULL, 那么属性页将加载由这个参数低位字部分所指定的字符串资源。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PropSheet_SetWizButtons

通过发送PSM_SETWIZBUTTONS消息, 来打开或关闭向导中的Back按钮、Next按钮与Finish按钮。可以使用这个宏, 也可以通过显式发送PSM_SETWIZBUTTONS消息的方法来完成此动作。

```
VOID PropSheet_SetWizButtons (
    HWND hPropSheetDlg,
    DWORD dwFlags
);
```

参数

hPropSheetDlg: 属性页的句柄。

dwFlags: 用来指明哪个向导按钮被打开的标志值, 可以使用以下一个或多个标志的组合:

PSWIZB_BACK	打开Back按钮。如果这个标志值没有被设置, 那么 Back 按钮将被显示为关闭状态。
PSWIZB_DISABLEDFINISH	显示一个关闭的Finish按钮。
PSWIZB_FINISH	显示一个打开的Finish按钮。
PSWIZB_NEXT	打开Next按钮。如果这个标志值没有被设置, 那么 Next 按钮将被显示为关闭状态。

返回值

没有返回值。

说明

这个宏将使用PostMessage来发送PSM_SETWIZBUTTONS消息。如果通告消息处理程序调用了PropSheet_SetWizButtons, 那么就不应该在处理程序返回之前执行任何影响窗口焦点的动作。例如, 如果在调用PropSheet_SetWizButtons之后立刻调用MessageBox, 那么消息框将接收到焦点。由于利用PostMessage所发送的消息需要直到到达消息队列的头部之后才能被传递; 因此, 其他消息将要直到向导失去了消息框焦点之后才被传递。这样, 属性页将不能正确地设置按钮的焦点。

向导将会在每个页面的底部显示三个或四个按钮, 这个消息则用来说明哪个按钮将要被打开。通常, 向导将显示Back按钮、Cancel按钮以及Next按钮或Finish按钮。典型情况下, 需要为欢迎页面打开Next按钮, 为内部页面打开Next按钮以及Back按钮, 为完成页面打开Back按钮与Finish按钮。而Cancel按钮则总是处于打开状态。一般地, 可以通过设置PSWIZB_FINISH或PSWIZB_DISABLEDFINISH来利用Finish按钮取代Next按钮。如果需要同时显示Next按钮与Finish按钮, 则应该在创建向导时, 在向导的PSH_WIZARDHASFINISHFLAG数据结构dwFlags数据成员中设置PROPSHEETHEADER。这样, 向导中的每个页面将显示四个按钮。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在prsht.h中进行了声明。

PropSheet_UnChanged

告知属性页, 某个页面中的信息已经恢复到先前保存的状态。可以使用这个宏, 也可以通过显式发送PSM_UNCHANGED消息的方法来完成此动作。

```
VOID PropSheet_UnChanged (hPropSheetDlg, hwndPage)
    HWND hPropSheetDlg,
    HWND hwndPage
);
```

参数

hPropSheetDlg: 属性页的句柄。

hwndPage: 已经恢复到先前保存状态的页面的句柄。

返回值

没有返回值。

说明

如果没有任何其他页面向属性页注册了改变信息, 属性页将关闭Apply Now按钮。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

21.4.4 属性页通告消息**PSN_APPLY**

这个通告消息用来告知应用程序, 用户已经单击了OK按钮、Close按钮或Apply按钮, 并且希望所有的改变生效。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_APPLY
    lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

如果需要阻止所做的改变生效, 就应该返回PSNRET_INVALID_NOCHANGEPAGE, 并且

将焦点返回给页面。如果接受所做的改变并且允许属性页被析构，就应该返回PSNRET_NOERROR。为了设置这个返回值，页面的对话框处理程序必须使用SetWindowLong函数（利用DWL_MSGRESULT值），并且对话框处理程序必须返回TRUE。

说明

如果用户单击了OK按钮，lppsn参数所指向的PSHNOTIFY数据结构中的lParam数据成员将为TRUE。此外，如果发送了PSM_CANCELTOCLOSE消息并且用户单击了Close按钮，也将会返回TRUE。而如果用户单击了Apply按钮，则将会返回FALSE。PSHNOTIFY数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。

在处理这个通告消息时，不应该调用EndDialog函数。

如果用户单击了OK按钮，并且应用程序在响应这个通告消息时返回了PSNRET_NOERROR值，那么这个属性页将被析构。

为了接受这个通告消息，页面必须在响应PSN_KILLACTIVE通告消息时将DWL_MSGRESULT值设置为FALSE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

PSN_GETOBJECT

当鼠标光标通过标签控件按钮的上方时，属性页将发送这个通告消息来请求拖放目标对象。

PSN_GETOBJECT

lpmmon = (LPNMOBJECTNOTIFY) lParam;

参数

lpmmon：NMOBJECTNOTIFY数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。如果这个通告消息被处理，就必须向这个数据结构中插入关于对象的信息。

返回值

处理这个通告消息的应用程序必须返回0。

说明

为了能够提供对象，应用程序必须在lpmmon所指向的NMOBJECTNOTIFY数据结构的某些数据成员中设置值。其中，pObject数据成员必须被设置为一个合法的对象指针，hResult数据成员必须被设置为一个成功标志。为了与COM标准保持一致，应该在提供对象指针时总是对对象的引用次数进行自加。

如果应用程序没有提供任何对象，那么必须将pObject数据成员设置为NULL，并且将hResult数据成员设置为一个失败标志。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

PSN_HELP

这个通告消息用来告知某个页面, 用户已经单击了Help按钮。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_HELP
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员中没有包含任何信息。

返回值

没有返回值。

说明

应用程序应该为页面显示帮助信息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSN_KILLACTIVE

由于另一个页面正处于激活状态或者用户单击了OK按钮, 这个通告消息将告知页面, 本页面将失去活跃状态。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_KILLACTIVE
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员中没有包

含任何信息。

返回值

如果需要阻止页面失去活跃状态，则返回TRUE；否则，返回FALSE。

说明

应用程序应该对用户所输入的信息进行合法性检查。

为了设置返回值，页面对话框处理程序必须使用SetWindowLong函数（带有DWL_MSGRESULT值），并且对话框处理程序必须返回TRUE。

如果对话框处理程序将DWL_MSGRESULT设置为TRUE，那么它应该显示一个消息框，用来向用户解释所存在的问题。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

PSN_QUERYCANCEL

这个通告消息用来告知应用程序，用户单击了Cancel按钮。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_QUERYCANCEL
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn：PSHNOTIFY数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的IPParam数据成员中没有包含任何信息。

返回值

如果需要阻止取消操作的发生，则应返回TRUE；否则，应该返回FALSE。

说明

属性页页面可以使用这个通告消息来向用户请求确信取消操作。

为了设置这个返回值，页面的对话框处理程序必须使用SetWindowLong函数（带有DWL_MSGRESULT值），并且对话框处理程序必须返回TRUE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

PSN_QUERYINITIALFOCUS

这个通告消息由属性页所发送，用来为属性页页面提供指定应该由哪个对话框控件接收初始焦点的方式。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_QUERYINITIALFOCUS
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的指针。应该为这个数据结构的lParam数据成员设置一个NMHDR类型的值，用来接收在缺省情况下将要被设置焦点的控件的句柄。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。

返回值

如果需要指定由哪个控件接收焦点，则应该返回控件的句柄。否则，应该返回0，这时焦点将被设置给缺省控件。如果需要设置返回值，对话框处理程序必须调用SetWindowLong函数（带有DWL_MSGRESULT值），并且返回TRUE。

示例

下面的代码段显示了PSN_QUERYINITIALFOCUS的一个简单处理程序实现，它要求初始焦点被设置给Location控件（IDC_LOCATION）。

```
case PSN_QUERYINITIALFOCUS:
    SetWindowLong (hDlg, DWL_MSGRESULT, (LPARAM) GetDlgItem (hDlg, IDC_LOCATION));
    return TRUE;
...
```

说明

在处理这个通告消息时，应用程序不应该调用SetFocus函数。同时，应该返回将要接收焦点的控件的句柄，并且属性页管理器将处理焦点的改变行为。

如果属性页管理器认为页面上没有任何控件应该接收焦点，那么将不会发送PSN_QUERYINITIALFOCUS通告消息。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PSN_RESET

这个通告消息用来告知页面，属性页将要被析构。这个通告消息是以WM_NOTIFY消息的

形式进行发送的。

```
PSN_RESET
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

没有返回值。

说明

自从最近一次PSN_APPLY通告消息被处理以来所做的所有改变都将被取消。如果用户单击了位于属性页右上角的“X”按钮, lppsn所指向的PSHNOTIFY数据结构的lParam数据成员将被设置为TRUE。如果用户单击了Cancel按钮, 那么将返回FALSE。PSHNOTIFY数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构, 这个NMHDR数据结构的hwndFrom数据成员必须包含属性页的句柄。

应用程序可以使用这个通告消息来执行清除操作。

在处理这个通告消息时, 不应该调用EndDialog函数。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSN_SETACTIVE

这个通告消息用来告知某个页面, 这个页面将要被激活。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_SETACTIVE
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员中没有包含任何信息。

返回值

如果需要接受页面激活动作, 则返回0; 如果需要激活后一个页面或前一个页面(取决于用户是单击了Next按钮还是单击了Back按钮), 则返回-1。如果需要将激活状态设置给某个特定的页面, 则返回这个页面的资源ID。

说明

必须在页面可见之前发送PSN_SETACTIVE通告消息。应用程序可以使用这个通告消息来初

始化页面中的数据。

如果需要设置返回值，那么这个页面的对话框处理程序就必须使用SetWindowLong函数（带有DWL_MSGRESULT值），并且对话框处理程序必须返回TRUE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在prsh.h中进行了声明。

参见

NMHDR。

PSN_TRANSLATEACCELERATOR

这个通告消息用来告知属性页，已经接收到一个键盘消息。这样就使得页面可以进行私有键盘加速转换。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_TRANSLATEACCELERATOR  
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn：PSHNOTIFY数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员是一个指向MSG数据结构的指针。这个数据成员可以是一个LPMSG数据类型值，用来获取将要被翻译的消息的参数。

返回值

如果返回PSNRET_MESSAGEHANDLED，则表示不需要进行任何进一步的处理；如果返回PSNRET_NOERROR，则表示需要请求进行正常的处理。

说明

为了设置返回值，这个页面的对话框处理程序必须调用SetWindowLong函数（带有DWL_MSGRESULT值），并且对话框处理程序必须返回TRUE。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 5或更新版本的Windows NT 4）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 5或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

PSN_WIZBACK

这个通告消息用来告知页面，用户已经单击了向导中的Back按钮。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_WIZBACK
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员中没有包含任何信息。

返回值

如果允许向导进入先前的页面，则返回0。如果需要阻止向导改变页面，则返回-1。如果需要显示特定的页面，则返回页面的对话框资源ID。

说明

为了设置返回值，页面的对话框处理程序必须调用SetWindowLong函数（带有DWL_MSGRESULT值），并返回TRUE。例如：

```
case PSN_WIZBACK;
    SetWindowLong (hDlg, DWL_MSGRESULT, 0);
    break;
case PSN_WIZNEXT;
...

```

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在prsht.h中进行了声明。

参见

PSN_WIZNEXT。

PSN_WIZFINISH

这个通告消息用来告知页面，用户已经单击了向导中的Finish按钮。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
PSN_WIZFINISH
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn: PSHNOTIFY数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的

hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员中没有包含任何信息。

返回值

- 如果需要阻止向导的完成，则返回TRUE。
- 在5.80版本有效。返回窗口句柄，用来阻止向导的完成。向导将把焦点设置给那个窗口。而且，窗口必须属于向导页面。
- 如果允许向导完成，则返回FALSE。

说明

为了设置返回值，页面的对话框处理程序必须使用SetWindowLong函数（带有DWL_MSGRESULT值），并且对话框处理程序必须返回TRUE。

在5.80版本有效。如果应用程序返回TRUE来阻止向导完成，那么它将不能控制页面中的那个窗口能够接收焦点。那些需要阻止向导完成的应用程序通常应该返回将要接收焦点的向导页面的窗口句柄，从而控制接收焦点的窗口。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在prsht.h中进行了声明。

PSN_WIZNEXT

这个通告消息用来告知页面，用户已经单击了向导中的Next按钮。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

PSN_WIZNEXT

```
lppsn = (LPPSHNOTIFY) lParam;
```

参数

lppsn：PSHNOTIFY数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。这个数据结构中包含有作为第一个数据成员hdr的NMHDR数据结构。而NMHDR数据结构的hwndFrom数据成员则包含有属性页的句柄。PSHNOTIFY数据结构的lParam数据成员中没有包含任何信息。

返回值

如果返回0，则表示允许向导进入下一个页面。如果返回-1，则表示将要阻止向导改变页面。如果需要显示一个特定的页面，则必须返回这个页面的对话框资源ID。

说明

为了设置返回值，页面的对话框处理程序必须调用SetWindowLong函数（带有DWL_MSGRESULT值），并返回TRUE。例如：

```
case PSN_WIZNEXT:
    SetWindowLong(hDlg, DWL_MSGRESULT, 0);
    break;
```

```
case PSN_WIZBACK;
...
```

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在prsh.h中进行了声明。

参见

PSN_WIZBACK。

21.4.5 属性页数据结构

PROPSHEETHEADER

这个数据结构定义了属性页的框架与页面。

```
typedef struct _PROPSHEETHEADER {
    DWORD dwSize;
    DWORD dwFlags;
    HWND hwndParent;
    HINSTANCE hInstance;
    union {
        HICON hIcon;
        LPCTSTR pszIcon;
    };
    LPCTSTR pszCaption;
    UINT nPages;
    union {
        UINT nStartPage;
        LPCTSTR pStartPage;
    };
    union {
        LPCPROPSHEETPAGE ppsp;
        HPROPSHEETPAGE FAR *phpage;
    };
    PFNPROPSHEETCALLBACK pfnCallback;

#ifdef _WIN32_IE_5_0
    union {
        HBITMAP hbmWatermark;
        LPCTSTR pszbmWatermark;
    };
    HPALETTE hplWatermark;
    union {
        HBITMAP hbmHeader;
        LPCTSTR pszbmHeader;
    };
#endif
}
```

```
};
#endif
}PROPSHEETHEADER, FAR *LPPROPSHEETHEADER;
```

成员

dwSize: 数据结构的大小, 单位为字节。属性页管理器利用这个数据成员来判断所使用的PROPSHEETHEADER数据结构的版本。关于更多信息, 请参见“说明”。

dwFlags: 这个数据成员用来说明在创建属性页页面时应该使用哪个选项。这个数据成员可以是下列值的组合:

PSH_DEFAULT: 对所有的数据结构数据成员使用缺省的含义。

PSH_HASHELP: 允许属性页页面显示Help按钮。在创建这个页面时, 也必须在页面的PROPSHEETPAGE数据结构中设置PSH_HASHELP标志。如果任何初始的属性页页面打开了一个Help按钮功能, 那么PSH_HASHELP将被自动地设置。如果没有任何初始页面打开了Help按钮功能, 则必须在那些希望拥有Help按钮的页面上特别地设置PSH_HASHELP。

PSH_HEADER: 在5.80版本中有效, 用来说明在Wizard97向导中将要使用头标的位图, 此外还必须设置PSH_WIZARD97标志位。除非设置了PSH_USEHBMHEADER标志位, 否则头标位图将从pszbmHeader数据成员中获取。如果设置了PSH_USEHBMHEADER标志位, 那么头标位图将从hbmHeader数据成员中获取。

PSH_MODELESS: 这个值将导致PropertySheet函数创建模态对话框形式的属性页, 而不是创建模态对话框形式的属性页。但这个标志位被设置之后, PropertySheet将在对话框被创建之后立刻返回, 并且从PropertySheet中的返回值是属性页对话框的窗口句柄。

PSH_NOAPPLYNOW: 删除Apply按钮。

PSH_NOCONTEXTHELP: 在5.80版本中有效, 用来删除上下文敏感的帮助按钮(“?”), 这个按钮通常出现在属性页的标题条上。这个标志值对于向导无效。关于在通用控件的早期版本中如何删除标题条帮助按钮的更多信息, 请参见第21章“属性页”。

PSH_PROPSHEETPAGE: 在创建属性页的页面时, 使用ppsp数据成员, 并忽略phpage数据成员。

PSH_PROPTITLE: 在属性页的标题条上, 紧跟在pszCaption数据成员所指定的字符串字后显示“Properties for”。

PSH_RTLLREADING: 获取pszCaption被显示的方向。常规的窗口将从左向右地(LTR)地显示包括pszCaption中的文本在内的所有文本。对于希伯来语或阿拉伯语之类的语言, 由于它们按照从右到左(RTL)的顺序进行读写, 因此必须对窗口进行镜像, 使得所有的文本以RTL的方式进行显示。如果设置了PSH_RTLLREADING, 那么在常规的父窗口中pszCaption中的文本应该按照RTL方式进行读写, 而在镜像父窗口中则应该按照LTR的方式进行读写。

PSH_STRETCHWATERMARK: 扩展与Internet Explorer 4.0相兼容的Wizard97样式向导的水印位图。

注意 这个样式主要是为了考虑对某些应用程序的相互兼容功能而设置的。因此, 并不推荐使用这个标志值, 并且这个标志值只在4.0版本与4.01版本的通用控件中提供了支持。

对于5.80版本及其后续版本的通用控件，这个标准值将被忽略。

PSH_USECALLBACK：在初始化由这个数据结构所定义的属性页时，调用由pfnCallback数据成员所指定的函数。

PSH_USEHBMHEADER：在5.80版本中有效，用来从hbmHeader数据成员而不是从pszbmHeader数据成员中获取头标位图。同时，必须设置PSH_WIZARD97与PSH_HEADER。

PSH_USEHBMWATERMARK：在5.80版本中有效，用来从hbmWatermark数据成员而不是从pszbmWatermark数据成员中获取水印位图。同时必须设置PSH_WIZARD97与PSH_WATERMARK。

PSH_USEHICON：使用hIcon作为属性页对话框中标题条的小图标。

PSH_USEHPLWATERMARK：在5.80版本中有效，使用由hplWatermark数据成员而不是由缺省的调色板所指向的HPALETTE数据结构来重绘水印位图与/或Wizard97向导的头标位图。同时必须设置PSH_WIZARD97与PSH_WATERMARK或PSH_HEADER。

PSH_USEICONID：使用pszIcon作为图标资源名进行加载，并且用它作为属性页对话框标题条的小图标。

PSH_USEPAGELANG：在5.80版本中有效，用来指明属性页所用的语言将从第一个页面资源中获取。

PSH_USESTARTPAGE：在显示属性页的初始页面时，使用pStartPage数据成员而不是使用nStartPage数据成员。

PSH_WATERMARK：在5.80版本中有效，用来说明水印位图将与Wizard97向导一同使用。同时必须设置PSH_WIZARD97标志位。除非设置了PSH_USEHBMWATERMARK，否则将从hbmWatermark数据成员中获取水印位图。在设置了PSH_USEHBMWATERMARK时，头标位图将从hbmWatermark数据成员中获取。

PSH_WIZARD：创建向导属性页。

PSH_WIZARD97：在5.80版本中有效，用来创建一个Wizard97样式的属性页，这个属性页中允许在其背景上显示头标位图与/或水印位图。

PSH_WIZARDCONTEXTHELP：添加一个上下文敏感的帮助按钮（“？”），这个帮助按钮一般是不会在向导的标题条中出现的。对于常规的属性页，这个标志位无效。

PSH_WIZARDHASFINISH：总是在向导中显示Finish按钮。同时必须设置PSH_WIZARD或PSH_WIZARD97。

PSH_WIZARD_LITE：在5.80版本中有效，用来表示使用Wizard样式。这种样式在外观上与PSH_WIZARD97相似，但它是由与PSH_WIZARD非常相似的方法进行实现的。对于如何进行页面的格式化，有几种限制。例如，不存在强制性边界、PSH_WIZARD_LITE样式不会像Wizard97样式那样绘制水印位图与头标位图。

hwndParent：属性页所有者窗口的句柄。

hInstance：从其中加载图标或标题字符串资源的实例句柄。如果pszIcon数据成员或pszCaption数据成员定义了将要被加载的资源，那么这个数据成员必须被指定。

hIcon：属性页对话框中用来作为标题条小图标的图标句柄。如果dwFlags数据成员中没有包

含PSH_USEHICON, 那么这个数据成员将被忽略。这个数据成员与pszIcon数据成员一同被声明为联合数据类型。

pszIcon: 属性页对话框中用来作为标题条小图标的图标资源。这个数据成员可以指定图标资源的ID, 或者图标资源名字符串的地址。如果dwFlags数据成员中没有包含PSH_USEICONID, 那么这个数据成员将被忽略。这个数据成员与hIcon数据成员一同被声明为联合数据类型。

pszCaption: 属性页对话框的标题。这个数据成员可以指定某个字符串资源的ID, 或者指定用来指明标题信息的字符串地址。如果dwFlags数据成员中包含有PSH_PROPTITLE, 那么字符串“Properties for”将被插入在标题的开始处。对于向导, 这个数据成员域将被忽略。

nPages: phpage数组中的元素数目。

nStartPage: 当属性页对话框被创建时, 所显示的初始页面基于0的索引。这个数据成员将与pStartPage一同被声明为联合数据类型。

pStartPage: 当属性页对话框被创建时, 所显示的初始页面名。这个数据成员可以指定字符串资源的ID, 或者指定表示资源名的字符串地址。这个数据成员将与nStartPage一同被声明为联合数据类型。

ppsp: PROPSHEETPAGE数据结构数组的地址, 这个数据结构定义了属性页中的页面。如果dwFlags数据成员中没有包含PSH_PROPSHEETPAGE, 那么这个数据成员将被忽略。需要说明的是, PROPSHEETPAGE数据结构的大小是可变的。那些对由ppsp数据成员所指向的数组进行解析的应用程序必须将每个页面大小考虑在内。这个数据成员将与phpage数据成员一同被声明为联合数据类型。

phpage: 属性页页面句柄所构成的数组的地址。数组中的每个句柄都必须被先前调用的CreatePropertySheetPage函数所创建。如果dwFlags数据成员中包含有PSH_PROPSHEETPAGE值, 那么phpage数据成员将被忽略并且应该被设置为NULL。当PropertySheet函数返回时, phpage数组中的任何HPROPSHEETPAGE句柄都将被析构。这个数据成员将与ppsp一同被声明为联合数据类型。

pfnCallback: 当属性页被初始化时, 被调用的应用程序所定义的信号回调函数地址。关于信号回调函数的更多信息, 请参见PropSheetProc函数的描述。如果dwFlags数据成员中没有包含PSH_USECALLBACK, 那么这个数据成员将被忽略。

hbmWatermark: 在5.80版本中有效, 用来表示水印位图的句柄。如果dwFlags数据成员没有包含PSH_USEHBMWATERMARK, 那么这个数据成员将被忽略。

pszbmWatermark: 在5.80版本中有效, 用来表示用作水印位图的位图资源。这个数据成员可以指定位图资源的ID或者位图资源名的字符串地址。如果dwFlags数据成员中没有包含PSH_USEHBMWATERMARK, 那么这个数据成员将被忽略。

hplWatermark: 在5.80版本中有效, 用来表示用于绘制水印位图与/或头标位图的HPALETTE数据结构。如果dwFlags数据成员中没有包含PSH_USEHPLWATERMARK, 那么这个数据成员将被忽略。

hbmHeader: 在5.80版本有效, 用来表示头标位图的句柄。如果dwFlags数据成员中没有包含PSH_USEHBMHEADER, 那么这个数据成员将被忽略。

pszbmHeader: 在5.80版本中有效,用来表示用作头标的位图资源。这个数据成员可以指定位图资源的ID或者指定位图资源名的字符串地址。如果dwFlags数据成员没有包含PSH_USEHBMHEADER,那么这个数据成员将被忽略。

说明

如果用户选择了将会扩大对话框的设置(例如大字体Large Fonts),那么页面与完成页面中所显示的水印位图也将会被扩大。而原始位图的大小与位置将仍然保持不变。这时,剩余的区域将被位图左上角的像素颜色填充。

需要说明的是,这个数据结构中的几个数据成员仅仅支持Comctl32.dll的4.71版本及其后续版本。可以在自己的头文件中添加以下一行代码来打开这些数据成员的使用:

```
#define WIN32_IE 0x0400// For version 4.71
```

或者

```
#define WIN32_IE 0x0500// For version 5.80
```

但是,必须用这个数据结构的大小对数据结构进行初始化。如果使用的是当前定义数据结构的大小,那么应用程序将可能不会运行于Comctl32.dll的早期版本,因为它们需要使用更小的数据结构。这种情况对于包括Microsoft Windows 95以及没有安装Internet Explorer 4.0及其后续版本的Microsoft Windows NT 4.0系统都会出现。可以通过定义应用程序版本号方法来运行4.71版本之前Comctl32.dll的应用程序。但是,如果应用程序需要运行于具有更新版本Comctl32.dll的系统中,那么就可能会导致错误。

可以使用当前头文件并且适当地设置PROPSHEETHEADER数据结构的大小,就可以实现所有Comctl32.dll版本的兼容性。在对这个数据结构进行初始化之前,应该使用DllGetVersion函数来判断自己计算机系统中安装的是哪个版本的Comctl32.dll。如果安装的是Comctl32.dll 4.70版本或更新版本,则应该使用以下语句:

```
psh.dwSize = sizeof( PROPSHEETHEADER );
```

来初始化dwSize数据成员。对于4.71版本之前的Comctl32.dll,这个数据结构由PROPSHEETHEADER_V1_SIZE常量给出,这时应该使用以下语句:

```
psh.dwSize = PROPSHEETHEADER_V1_SIZE;
```

三种向导样式PSH_WIZARD、PSH_WIZARD97与PSH_WIZARD_LITE样式在操作上是互不兼容的。在同一时刻只能设置其中的一种样式。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsht.h中进行了声明。

PROPSHEETPAGE

这个数据结构定义了属性页中的一个页面。

```

typedef struct _PROPSHEETPAGE {
    DWORD dwSize;
    DWORD dwFlags;
    HINSTANCE hInstance;
    union {
        LPCSTR pszTemplate;
        LPCTSTR pResource;
    };
    union {
        HICON hIcon;
        LPCSTR pszIcon;
    };
    LPCSTR pszTitle;
    DLGPROC pfnDlgProc;
    LPARAM lParam;
    LPPNPSPCALLBACK pfnCallback;
    UINT FAR *pRefParent;

    #if (_WIN32_IE >= 0x0500)
        LPCTSTR pszHeaderTitle;
        LPCTSTR pszHeaderSubTitle;
    #endif
} PROPSHEETPAGE, FAR *LPPROPSHEETPAGE;

```

成员

dwSize: 这个数据结构的大小, 单位为字节。属性页管理器使用这个数据成员来判断正在使用的是哪个版本的PROPSHEETHEADER数据结构。

dwFlags: 这个数据成员用来说明在创建属性页页面时所使用的选项, 它可以是以下值的组合:

PSP_DEFAULT: 对所有的数据结构数据成员使用缺省含义。

PSP_DLGINDIRECT: 从pResource数据成员所指向的内存中的对话框模板创建页面。PropertySheet函数假设内存中的这个模板没有被写保护。如果使用的是只读模板, 在某些版本的Microsoft Windows中可能会导致意外中断。

PSP_HASHELP: 当页面处于活跃状态时, 打开属性页的Help按钮。

PSP_HIDEHEADER: 在5.80版本有效, 设置这个值, 将使得在页面被选中时, 向导属性页隐藏标题区域。如果提供了水印位图, 那么这个水印位图将在页面的左边被绘制。应该为欢迎页面与完成页面设置这个标志位, 而在内部页面中应该取消这个标志位。

PSP_PREMATURE: 在4.71版本有效, 设置这个值, 将使得在属性页被创建的同时以创建一个页面。如果没有设置这个标志位, 页面将会直到它第一次被选中之后才创建。

PSPRTLREADING: 按照相反的方向显示pszTitle中的内容。通常, 窗口是按照从左到右(LTR)的形式显示包括pszTitle在内的所有文本的。对于希伯来语或阿拉伯语之类的语言, 由于它们是按照从右到左(RTL)的顺序进行读写, 因此必须对窗口进行镜像, 使得所有的文本按照RTL方式进行显示。如果设置了PSPRTLREADING, 那么在常规的父窗口中pszTitle将按照RTL

的方式进行显示，而在镜像父窗口中则按照LTR pszTitle进行显示。

PSP_USECALLBACK：在创建或析构由这个数据结构所定义的属性页页面时，调用由pfnCallback数据成员所指定的函数。

PSP_USEHEADERSUBTITLE：在5.80版本中有效，如果设置了这个值，将把由pszHeaderSubTitle数据成员所指向的字符串当作Wizard97页面头标区域的子标题进行显示。为了使得这个标志位有效，必须在与向导相关联的PROPSHEETHEADER数据结构dwFlags数据成员中设置PSH_WIZARD97标志位。如果设置了PSP_HIDEHEADER，那么PSP_USEHEADERSUBTITLE标志位将被忽略。

PSP_USEHEADERTITLE：在5.80版本中有效，如果设置了这个值，将把由pszHeaderTitle数据成员所指向的字符串当作Wizard97内部页面头标的标题进行显示。必须在与向导相关联的PROPSHEETHEADER数据结构dwFlags数据成员中设置PSH_WIZARD97标志位。如果设置了PSP_HIDEHEADER，那么PSP_USEHEADERTITLE标志位将被忽略。

PSP_USEHICON：使用hIcon作为页面标签的小图标。

PSP_USEICONID：使用pszIcon作为将要被加载的图标资源名，并且用它作为页面标签的小图标。

PSP_USEREFPARENT：在利用这个数据结构所创建的属性页页面生命周期内，维护由pcRefParent数据成员所指定的引用计数。

PSP_USETITLE：使用pszTitle数据成员作为属性页对话框的标题，而不是用存储在对话框模板中的信息作为标题。

hInstance：将要从其中加载图标或字符串资源的实例句柄。如果pszIcon、pszTitle、pszHeaderTitle或pszHeaderSubTitle数据成员定义了将要加载的资源ID，那么必须指定hInstance。

pszTemplate：用来创建页面的对话框模板。这个数据成员可以指定模板的资源ID或指定模板名的字符串地址。如果dwFlags数据成员中的PSP_DLGINDDIRECT标志位被设置，那么pszTemplate将被忽略。这个数据成员将与pResource数据成员一同被声明为联合数据类型。

pResource：内存中对话框模板的地址。PropertySheet函数假设这个模板是没有被写保护的。在某些版本的Windows中，只读模板将可能会导致意外中断。为了使用这个数据成员，必须设置dwFlags数据成员中的PSP_DLGINDDIRECT标志位。这个数据成员将与pszTemplate数据成员一同被声明为联合数据类型。

hIcon：将要被用做页面标签中图标的标签句柄。如果dwFlags数据成员中没有包含PSP_USEHICON，那么这个数据成员将被忽略。这个数据成员将与pszIcon数据成员一同被声明为联合数据类型。

pszIcon：将被用做页面标签中图标的图标资源。这个数据成员可以指定图标资源的ID或指定用来标识图标资源名的字符串地址。如果需要使用这个数据成员，必须在dwFlags数据成员中设置PSP_USEICONID标志位。这个数据成员将与hIcon数据成员一同被声明为联合数据类型。

pszTitle：属性页对话框的标题。这个标题将覆盖对话框模板中所指定的标题。这个数据成

员可以指定字符串资源的ID或用来表明这个标题的字符串地址。为了使用这个数据成员，必须在dwFlags数据成员中设置PSH_USESTITLE标志位。

pfnDlgProc: 页面对话框处理程序的地址。由于这个页面被创建为非模态对话框的形式，因此，处理程序必须不应该调用EndDialog函数。

lParam: 在页面被创建之后，将利用WM_INITDIALOG消息向对话框处理程序传递这个页面PROPSHEETPAGE数据结构的一个副本。而lParam数据成员可用来向对话框处理程序传递与特定应用程序相关的信息。这个数据成员对于页面自身没有任何影响。关于更多信息，请参见21.1.3节“创建属性页”。

pfnCallback: 应用程序定义的信号回调函数地址，这个函数在页面被创建时以及在页面将要被析构时被调用。关于信号回调函数的更多信息，请参见PropSheetPageProc。如果需要使用这个数据成员，必须在dwFlags数据成员中设置PSP_USECALLBACK标志位。

pcRefParent: 引用计数值的地址。如果需要用这个数据成员，必须在dwFlags数据成员中设置PSP_USEREFPARENT标志位。

注意 在创建属性页页面时，由pcRefParent所指向的值将会自增。如果设置了PROPSHEETHEADER数据结构dwFlags数据成员中的PSH_PROPSHEETPAGE标志位，并且调用了PropertySheet函数，那么也就隐晦地创建了一个属性页页面。当然，也可以使用CreatePropertySheetPage函数来明确地完成这个动作。当一个属性页页面被析构时，由pcRefParent数据成员所指向的值将自减，这个动作在属性页被析构时将自动地发生。当然，也可以使用DestroyPropertySheetPage函数来明确地析构一个属性页页面。

pszHeaderTitle: 在5.80版本中有效，用来表示头标区域的标题。为了使用这个数据成员，必须：

- 在dwFlags数据成员中设置PSP_USEHEADERSUBTITLE标志位。
- 在页面PROPSHEETHEADER数据结构dwFlags数据成员中设置PSH_WIZARD97标志位。
- 确信dwFlags数据成员中的PSP_HIDEHEADER标志位没有被设置。

pszHeaderSubTitle: 在5.80版本中有效，用来表示头标区域的子标题。为了使用这个数据成员，必须：

- 在dwFlags数据成员中设置PSP_USEHEADERSUBTITLE标志位。
- 在页面的PROPSHEETHEADER数据结构dwFlags数据成员中设置PSH_WIZARD97标志位。
- 确信dwFlags数据成员PSP_HIDEHEADER标志位没有被设置。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。

PSHNOTIFY

这个数据结构中包含有关于属性页通告消息的信息。

```
typedef struct_PSHNOTIFY {  
    NMHDR hdr;  
    LPARAM lParam;  
}PSHNOTIFY, FAR *LPPSHNOTIFY;
```

成员

hdr: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

lParam: 这种数据成员中包含有关于本通告消息的更多信息。如果需要判断在这个数据成员中存放的是哪些信息(若存在), 请参见特定通告消息的描述。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在prsh.h中进行了声明。



第22章 Rebar 控件

Rebar控件用来作为子窗口的容器。应用程序可以将子窗口（通常为其他的控件）分配给Rebar控件区段。Rebar控件中包含有一个或多个区段，而每个区段都可以是夹头条、位图、文本标签与子窗口的组合。但是，区段中不能包含多个子窗口。

22.1 关于Rebar控件

Rebar控件在指定背景位图之上显示子窗口。当用户动态改变Rebar控件区段的位置时，Rebar控件也将对分配给这个区段的子窗口的大小与位置进行管理。

图22-1演示了包含有两个区段的Rebar控件。其中一个区段包含有组合框，另一个区段则包含有透明的工具条控件。

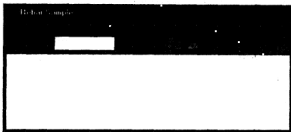


图22-1 Rebar控件

注意 Rebar控件在Comctl32.dll的4.70版本及其后续版本中被实现。

22.1.1 Rebar区段和子窗口

应用程序使用RB_INSERTBAND消息与RB_SETBANDINFO消息来定义Rebar控件区段的性状。这些消息能够接受于REBARBANDINFO数据结构的地地址作为lParam数据成员。而REBARBANDINFO数据结构的数据成员则定义了给定区段的性状。如果需要设置某个区段的性状，应该将cbSize数据成员设置为表明这个数据结构的大小（单位为字节）。然后，将fMask数据成员设置为表明应用程序将要填充哪个数据成员。

如果需要将一个子窗口分配给区段，那么应该在REBARBANDINFO数据结构的数据成员中包含RBBIM_CHILD标志位，然后将hwndChild数据成员设置为子窗口的句柄。应用程序可以分别在cxMinChild数据成员与cyMinChild数据成员中设置子窗口的最小允许宽度与高度。

当Rebar控件被析构时，它将析构分配给区段中的任何子窗口。为了防止Rebar控件析构分配给其区段中的子窗口，应该发送RB_DELETEBAND消息来删除这个区段，然后在析构Rebar控

件之前利用SetParent函数将父窗口重新设置给另一个窗口。

22.1.2 Rebar控件用户界面

除了那些使用RBBS_FIXEDSIZE样式的Rebar控件区段不能改变大小之外，其他所有的Rebar控件区段都可以改变大小。如果需要改变Rebar控件中区段的大小或改变这些区段的出现顺序，就应该单击并拖动区段的夹头条。Rebar控件将自动地对分配给区段的子窗口进行大小调整与位置调整。此外，可以在区段的文本（若存在）上进行单击，从而切换地改变区段的大小。

22.1.3 Rebar控件图像列表

如果应用程序正在对Rebar控件使用图像列表，那么它必须在将区段添加到控件中之前发送RB_SETBARINFO消息。这个消息能够接受REBARINFO数据结构的地址作为lParam参数。在发送这个消息之前，应该将cbSize数据成员设置为REBARINFO数据结构的大小（单位字节），从而准备这个数据结构。然后，如果Rebar控件将要在区段上显示图像，那么就应该将fMask数据成员设置为RBIM_IMAGELIST标志，并且将一个图像列表的句柄分配给hIml数据成员。如果Rebar控件没有使用任何区段图像，那么应该将fMask数据成员设置为0。

22.1.4 Rebar控件消息转发

Rebar控件能够将所有的WM_NOTIFY窗口消息转发给自己的父窗口。此外，Rebar控件还能转发任何从窗口发送给区段的消息（例如WM_CHARTOITEM消息、WM_COMMAND消息等等）。

22.1.5 支持定制绘图

Rebar控件能够支持定制绘图功能。关于这方面的更多信息，请参见8.1节“关于定制绘图”。

22.2 使用Rebar控件

本节将给出一段示例代码，用来演示如何实现Rebar控件。

创建Rebar控件

应用程序调用CreateWindowEx函数，同时把REBARCLASSNAME指定为窗口类，从而创建Rebar控件。为此，应用程序首先必须调用InitCommonControlsEx函数，并且同时设置INITCOMMONCONTROLSEX数据结构中的ICC_COOL_CLASSES位，从而对这个窗口类进行注册。

下面的例子将创建带有两个区段的Rebar控件，其中一个区段包含有组合框，另一个区段包含有一个工具条。在这个例子中使用了RBS_VARHEIGHT样式，从而允许控件使用可变的区段高度。在创建Rebar控件之后，CreateRebar将调用两个由应用程序所定义的函数（CreateComboBox函数与CreateToolBar函数），从而创建子窗口。在添加每个区段之前，CreateRebar将初始化REBARBANDINFO数据结构中的cbSize数据成员，这与RB_INSERTBAND消

息所要求的一样。然后，将设置数据结构的fMask数据成员以反映哪个数据成员包含有合法的数据。CreateRebar为每个区段设置cyMinChild数据成员，从而允许区段中控件的高度信息有效。如果cxMinChild数据成员的值为0，那么就表示允许用户完全在给定的区段中隐藏这个控件。

```

HWND WINAPI CreateRebar(HWND hwndOwner)
{
    REBARINFO        rbi;
    REBARBANDINFO     rbBand;
    RECT             rc;
    HWND hwndCB, hwndTB, hwndRB;
    DWORD dwBtnSize;
    INITCOMMONCONTROLSEX icex;

    icex.dwSize = sizeof(INITCOMMONCONTROLSEX);
    icex.dwICC = ICC_COOL_CLASSES | ICC_BAR_CLASSES;
    InitCommonControlsEx(&icex);

    hwndRB = CreateWindowEx(WS_EX_TOOLWINDOW,
                           REBARCLASSNAME,
                           NULL,
                           WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS |
                           WS_CLIPCHILDREN | RBS_VARHEIGHT | CCS_NODIVIDER,
                           0, 0, 0, 0,
                           hwndOwner,
                           NULL,
                           g_hinst,
                           NULL);

    if(!hwndRB)
        return NULL;

    //Initialize and send the REBARINFO structure.
    rbi.cbSize = sizeof(REBARINFO); //Required when using this struct.
    rbi.fMask = 0;
    rbi.himl = (HIMAGELIST) NULL;
    if(!SendMessage(hwndRB, RB_SETBARINFO, 0, (LPARAM) &rbi))
        return NULL;

    //Initialize structure members that both bands will share.
    rbBand.cbSize = sizeof(REBARBANDINFO); //Required
    rbBand.fMask = RBBIM_COLORS | RBBIM_TEXT | RBBIM_BACKGROUND |
                  RBBIM_STYLE | RBBIM_CHILD | RBBIM_CHILDSIZE |
                  RBBIM_SIZE;
    rbBand.fStyle = RBBS_CHILDEDGE | RBBS_FIXEDBMP;
    rbBand.hbmBack = LoadBitmap(g_hinst, MAKEINTRESOURCE(IDB_BACKGRND));

    //Create the combo-box control to be added.
    hwndCB = CreateComboBox(hwndRB);

```

```

//Set values unique to the band with the combo box.
GetWindowRect(hwndCB,&rc);
rbBand.lpText      ="Combo Box";
rbBand.hwndChild   =hwndCB;
rbBand.cxMinChild  =0;
rbBand.cyMinChild  =rc.bottom -rc.top;
rbBand.cx          =200;

//Add the band that has the combo box.
SendMessage(hwndRB,RB_INSERTBAND,(WPARAM)-1,(LPARAM)&rbBand);

//Create the tool bar control to be added.
hwndTB =CreateToolBar(hwndOwner,dwStyle);

//Get the height of the tool bar .
dwBtnSize =SendMessage(hwndTB,TB_GETBUTTONSIZE,0,0);

//Set values unique to the band with the tool bar .
rbBand.lpText      ="Tool Bar";
rbBand.hwndChild   =hwndTB;
rbBand.cxMinChild  =0;
rbBand.cyMinChild  =HIWORD(dwBtnSize);
rbBand.cx          =250;

//Add the band that has the tool bar .
SendMessage(hwndRB,RB_INSERTBAND,(WPARAM)-1,(LPARAM)&rbBand);

return (hwndRB);
}

```

22.3 Rebar控件样式

除了标准的窗口样式，Rebar控件还支持许多其他的控件样式。

RBS_AUTOSIZE

在4.71版本中有效。在这个样式下，当控件的大小或位置发生改变时，Rebar控件将自动地改变区段的布局。这时，将发送RBN_AUTOSIZE通告消息。

RBS_BANDBORDERS

在4.70版本中有效。在这个样式下，Rebar控件将显示狭窄的线条用来分开相连的区段。

RBS_DBLCLKTOGGLE

在4.71版本中有效。在这个样式下，当用户双击区段时，Rebar控件区段将在自己的最大化状态与最小化状态之间切换。如果没有使用这个样式，那么当用户单击区段时，Rebar控件区段将在自己的最大化状态与最小化状态之间进行切换。

RBS_FIXEDORDER

在4.70版本中有效。在这个样式下，Rebar控件总是按照

RBS_REGISTERDROP

相同的顺序显示区段。用户可以将区段移动到不同的行上，但区段的顺序是静止不变的。

在4.71版本中有效。在这个样式下，当某个对象被拖动到Rebar控件的一个区段上时，Rebar控件将产生RBN_GETOBJECT通告消息。为了能够接收RBN_GETOBJECT通告消息，必须调用OleInitialize函数或CoInitialize函数来初始化OLE。

RBS_TOOLTIPS

在4.70版本中有效。这个样式尚未提供支持。

RBS_VARHEIGHT

在4.70版本中有效。在这个样式下，只要有可能，Rebar控件将按照最小的高度显示区段。如果没有使用这个样式，Rebar控件将利用相同的宽度来显示所有的区段，并且采用最高可见区段的高度来决定其他区段的高度。

RBS_VERTICALGRIPPER

在4.71版本中有效。在这种样式下，在竖直Rebar控件中，大小夹具条将被竖直地进行显示，而不是水平地显示。对于没有使用CCS_VERT样式的Rebar控件，这个样式将被忽略。

22.4 Rebar控件参考

22.4.1 Rebar控件消息

RB_BEGINDRAG

将Rebar控件设置为拖放模式。这个消息不会导致RBN_BEGINDRAG通告消息被发送。

RB_BEGINDRAG

wParam = (WPARAM)(UINT) uBand;

lParam = (LPARAM)(DWORD) dwPos;

参数

uBand: 拖放操作将要影响区段的基于0的索引。

dwPos: 包含有鼠标起始坐标的DWORD值。其中，鼠标的水平坐标包含在LOWORD中，而竖直坐标包含在HIWORD中。如果传递了(DWORD)-1，那么Rebar控件将使用控件的线程最后一次调用GetMessage或PeekMessage时所使用的鼠标位置。

返回值

这个消息的返回值没有被使用。

说明

RB_BEGINDRAG消息、RB_DRAGMOVE消息与RB_ENDDRAG消息可以用来为Rebar控件实现IDropTarget界面。在响应IDropTarget::DragEnter时，应该发送RB_BEGINDRAG消息；在响应IDropTarget::DragOver时，应该发送RB_DRAGMOVE消息；在响应IDropTarget::Drop与

IDropTarget::DragLeave时, 应该发送RB_ENDDRAG消息。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RB_DELETEBAND

从Rebar控件中删除一个区段。

RB_DELETEBAND

```
wParam = (WPARAM)(UINT) uBand;
lParam = 0;
```

参数

uBand: 将要被删除区段的基于0的索引。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_DRAGMOVE

在处理先前的RB_BEGINDRAG消息之后, 更新Rebar控件拖放位置。

RB_DRAGMOVE

```
wParam = 0;
lParam = (LPARAM)(DWORD) dwPos;
```

参数

dwPos: 包含有鼠标起始坐标的DWORD值。其中, 鼠标的水平坐标包含在LOWORD中, 而竖直坐标包含在HIWORD中。如果传递了(DWORD)-1, 那么Rebar控件将使用控件的线程最后一次调用GetMessage或PeekMessage时所使用的鼠标位置。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_ENDDRAG。

RB_ENDDRAG

结束Rebar控件的拖放操作。这个消息并不会导致RBN_ENDDRAG通告消息被发送。

```
RB_ENDDRAG
    wParam = 0;
    lParam = 0;
```

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_BEGINDRAG、RB_DRAGMOVE。

RB_GETBANDBORDERS

获取区段的边界, 这个消息所得到的结果可用来计算区段的可使用区域。

```
RB_GETBANDBORDERS
    wParam = (WPARAM)(UINT) uBand;
    lParam = (LPARAM)(LPRECT) lprc;
```

参数

uBand: 将要被计算其边界的区段基于0的索引。

lprc: 将要接收区段边界的RECT数据结构的地址。如果Rebar控件使用了RBS_

BANDBORDERS样式, 那么这个数据结构的每个数据成员都将接收到构成区段边界每一边的相应像素数。如果Rebar控件没有使用RBS_BANDBORDERS样式, 那么只有这个数据结构的left数据成员将会接收到合法的信息。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_GETBANDCOUNT

获取Rebar控件中当前区段的数目。

RB_GETBANDCOUNT

wParam = 0;

lParam = 0;

返回值

返回表示被分配给Rebar控件的区段数目的UINT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_GETBANDINFO

获取关于Rebar控件中指定区段的信息。

RB_GETBANDINFO

wParam = (WPARAM)(UINT) uBand;

lParam = (LPARAM)(LPREBARBANDINFO) lprbbi;

参数

uBand: 将要从其中获取信息的区段基于0的索引。

lprbbi: 将要接收所请求信息的REBARBANDINFO数据结构的地址。在发送这个消息之前,

必须将这个数据结构的cbSize数据成员设置为REBARBANDINFO数据结构大小，并且将fMask参数设置为将要获取其信息的项目。此外，必须将REBARBANDINFO数据结构的cch数据成员设置为lpText缓冲器的大小（在RBBIM_TEXT被指定时）。

返回值

若操作成功，则返回一个非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 3.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

RB_SETBANDINFO。

RB_GETBARHEIGHT

获取Rebar控件的高度信息。

RB_GETBARHEIGHT

wParam = 0;

lParam = 0;

返回值

返回代表Rebar控件高度（单位为像素）的UINT值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

RB_GETBARINFO

获取关于Rebar控件信息以及Rebar控件所使用的图像列表信息。

RB_GETBARINFO

wParam = 0;

lParam = (LPARAM)(LPREBARINFO) lprbi;

参数

lprbi: 将要接收Rebar控件信息的REBARINFO数据结构的地址。在发送这个消息之前, 应该将这个数据结构的cbSize数据成员设置为sizeof(REBARINFO) (也就是REBARINFO数据结构的大小)

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_GETBKCOLOR

获取Rebar控件的缺省背景色。

```
RB_GETBKCOLOR
    wParam = 0;
    lParam = 0;
```

返回值

返回用来表示Rebar控件当前缺省背景色的COLORREF值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

RB_SETBKCOLOR。

RB_GETCOLORSCHEME

从Rebar控件中获取颜色方案信息。

```
RB_GETCOLORSCHEME
    wParam = 0;
    lParam = (LPARAM)(LPCOLORSCHEME) lpcs;
```

参数

lpcs: 将要接收颜色方案信息的COLORSCHEME数据结构的地址。在发送这个消息之前, 必须将数据结构的dwSize数据成员设置为sizeof(COLORSCHEME) (也就是COLORSCHEME数据结构的大小)。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_SETCOLORSCHEME。

RB_GETDROPTARGET

获取Rebar控件的IDropTarget接口指针。

RB_GETDROPTARGET

wParam = 0;

lParam = (IDropTarget**) ppDropTarget;

参数

ppDropTarget: 将要获取接口指针的IDropTarget指针地址。当指针不再需要时, 程序调用者将负责调用这个指针Release函数来释放这个指针。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RB_GETPALETTE

获取Rebar控件的当前调色板。

```
RB_GETPALETTE
    wParam = 0;
    lParam = 0;
```

返回值

返回用来指明Rebar控件当前调色板的HPALETTE值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_SETPALETTE。

RB_GETRECT

获取Rebar控件中给定区段的约束矩形。

```
RB_GETRECT
    wParam = (WPARAM)(INT) iBand;
    lParam = (LPARAM)(LPRECT) lprc;
```

参数

iBand: Rebar控件中区段基于0的索引。

lprc: 将要接收Rebar控件区段约束矩形的RECT数据结构地址。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_GETROWCOUNT

获取Rebar控件中区段所占用的行数。

```
RB_GETROWCOUNT
```

```
wParam = 0;
lParam = 0;
```

返回值

返回用来表示Rebar控件中区段行数的UINT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_GETROWHEIGHT

获取Rebar控件中指定行的高度。

```
RB_GETROWHEIGHT
wParam = (WPARAM)(UINT) uRow;
lParam = 0;
```

参数

uRow: 区段基于0的索引。包含有给定区段的行的高度将被获取。

返回值

返回来表示行高度 (单位为像素) 的UINT值。

说明

如果需要获取Rebar控件中的行数, 应该使用RB_GETROWCOUNT消息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_GETTEXTCOLOR

获取Rebar控件的缺省文本颜色。

```
RB_GETTEXTCOLOR
wParam = 0;
lParam = 0;
```

返回值

返回用来表示当前缺省文本颜色的COLORREF值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

RB_SETTEXTCOLOR。

RB_GETTOOLTIPS

获取被分配给Rebar控件中任何工具提示控件的句柄。

```
RB_GETTOOLTIPS  
wParam = 0;  
lParam = 0;
```

返回值

返回与Rebar控件相关联的工具提示控件句柄的HWND值; 或者如果没有任何工具提示控件与Rebar控件相关联, 则返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RB_GETUNICODEFORMAT

获取Rebar控件UNICODE字符格式标志。

```
RB_GETUNICODEFORMAT  
wParam = 0;  
lParam = 0;
```

返回值

返回Rebar控件的UNICODE格式标志。如果这个值为非零, 那么表示控件正在使用UNICODE字符; 如果这个值为0, 那么表示控件正在使用ANSI字符。

说明

关于这个消息的更多讨论, 请参见CCM_GETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_SETUNICODEFORMAT。

RB_HITTEST

判断Rebar区段的哪个部分正在屏幕上的给定点 (如果Rebar区段在这个点上存在)。

RB_HITTEST

wParam = 0;

lParam = (LPARAM)(LPRBHITTESTINFO) lprbht;

参数

lprbht: RBHITTESTINFO数据结构的地址。在发送这个消息之前, 这个数据结构的pt数据成员必须被初始化为将要被进行鼠标击中测试的点 (在客户区坐标中)。

返回值

返回给定点上的区段基于0的索引, 或者, 如果在给定点上不存在任何Rebar区段, 则返回-1。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_IDTOINDEX

将区段标识符转换为Rebar控件中的区段索引。

RB_IDTOINDEX

wParam = (WPARAM)(UINT) uBandID;

lParam = 0;

参数

uBandID: 正在处理的区段中有应用程序所定义的标识符。这个参数是在区段被插入时, 被传递给REBARBANDINFO数据结构的wID数据成员的值。

返回值

如果操作成功, 则返回这个区段基于0的索引; 否则, 返回-1。如果存在区段标识符同名的现象, 那么将返回第一个区段的索引。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_INSERTBAND

向Rebar控件中插入一个新的区段。

RB_INSERTBAND

```
wParam = (WPARAM)(UNIT) uIndex;  
lParam = (LPARAM)(LPREBARBANDINFO) lprbbi;
```

参数

uIndex: 区段将要被插入位置基于0的索引。如果将这个参数设置为-1, 那么控件将把新区段添加在Rebar控件的最后。

lprbbi: 用来定义将要被插入的区段的REBARBANDINFO数据结构地址。在发送这个消息之前, 必须将这个数据结构的cbSize数据成员设置为sizeof(REBARBANDINFO) (也就是REBARBANDINFO数据结构的大小)。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_MAXIMIZEBAND

将Rebar控件中某个区段的大小设置为理想大小或者最大大小。

```
RB_MAXIMIZEBAND
    wParam = (WPARAM)(UINT) uBand;
    lParam = (LPARAM)(BOOL) fIdeal;
```

参数

uBand: 将要被最大化的区段基于0的索引。

fIdeal: 这个参数用来说明在区段内最大化时是否要使用区段的理想宽度。如果这个值为非零, 那么表示将要使用理想宽度。如果这个值为0, 那么表示区段将被设置得尽可能大。

返回值

所得的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_MINIMIZEBAND

将Rebar控件中某个区段的大小设置为最小大小。

```
RB_MINIMIZEBAND
    wParam = (WPARAM)(UINT) uBand;
    lParam = 0;
```

参数

uBand: 将要被最小化的区段基于0的索引。

返回值

所得到的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_MOVEBAND

将区段从一个索引转移到另一个索引。

```
RB_MOVEBAND
    wParam = (WPARAM)(UINT) iFrom;
    lParam = (LPARAM)(UINT) iTo;
```

参数

iFrom: 将要被转移的区段基于0的索引。

iTo: 新区段位置基于0的索引。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

说明

这个消息很有可能会改变Rebar控件中其他区段的索引。如果某个区段的索引从6移动到0, 那么位于这两个索引之间的所有区段的索引都将会自增1。

iTo参数必须不能大于Rebar控件中区段的数目减1。可以使用RB_GETBANDCOUNT消息来获取Rebar控件中的区段数目。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RB_PUSHCHEVRON

这个消息被发送给Rebar控件, 用来通过编程的方法推动一个人字形。

```
RB_PUSHCHEVRON
    wParam = (WPARAM)(UINT) uBand;
    lParam = (LPARAM) lParamValue;
```

参数

uBand: 其人字形将要被推动的区段基于0的索引。

lParamValue: 由应用程序所定义的32位值。当NMREBARCHEVRON数据结构的lParamNM数成员被RBN_CHEVRONPUSHED通告消息传递时, 这个参数将被传递给应用程序。

返回值

这个通告消息的返回值没有被使用。

说明

当这个消息被发送时, Rebar控件将向应用程序发送一个RBN_CHEVRONPUSHED通告消息, 请求应用程序显示人字形菜单。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RB_SETBANDINFO

设置Rebar控件中现有区段的特性信息。

RB_SETBANDINFO

wParam = (WPARAM)(UINT) uBand;

lParam = (LPARAM)(LPREBARBANDINFO) lprbbi;

参数

uBand: 将要接收新设置的区段基于0的索引。

lprbbi: REBARBANDINFO数据结构的地址, 这个数据结构用来定义将要被修改的区段与新设置信息。在发送这个消息之前, 必须将这个数据结构的cbSize数据成员设置为sizeof (REBARBANDINFO) (也就是REBARBANDINFO数据结构的大小)。此外, 必须将REBARBANDINFO数据结构的cch数据成员设置为lpText缓冲器的大小 (如果RBBIM_TEXT被指定)。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_SETBARINFO

设置Rebar控件特性信息。

RB_SETBARINFO

wParam = 0;

lParam = (LPARAM)(LPREBARINFO) lprbi;

参数

lprbi: REBARINFO数据结构的地址, 在这个数据结构中包含有将要被设置的信息。在发送这个消息之前, 必须将这个数据结构的cbSize数据成员设置为sizeof (REBARBANDINFO) (也

就是REBARBANDINFO数据结构的大小)。

返回值

若操作成功,则返回一个非零值;否则,返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_SETBKCOLOR

设置Rebar控件的缺省背景色。

RB_SETBKCOLOR

```
wParam = 0;  
lParam = (LPARAM)(COLORREF) clrBk;
```

参数

clrBk: 代表新缺省背景色的COLORREF值。

返回值

返回代表先前缺省背景色的COLORREF值。

说明

Rebar控件缺省背景色将用来绘制Rebar控件的背景以及在这个消息被发送之后添加到其Rebar控件中的所有区段的背景。当有某个区段被添加,或者被通过设置REBARBANDINFO数据结构fMask数据成员的RBBIM_COLORS标志位以及设置clrBack数据成员而进行修改时,某个特定区段的缺省背景色将可能会被覆盖。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

RB_GETBKCOLOR。

RB_SETCOLORSCHEME

设置Rebar控件的颜色方案信息。

```
RB_SETCOLORSCHEME  
wParam = 0;  
lParam = (LPARAM)(LPCOLORSCHEME) lpcls;
```

参数

lpcls: COLORSCHEME数据结构的地址, 在这个数据结构中包含有颜色方案信息。

返回值

这个消息返回值没有被使用。

说明

Rebar控件在绘制控件中以及区段中的三维元素时将使用颜色方案信息。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_GETCOLORSCHEME。

RB_SETPALETTE

设置Rebar控件的当前调色板。

```
RB_SETPALETTE  
wParam = 0;  
lParam = (LPARAM)(HPALETTE) hpal;
```

参数

hpal: 用来指明Rebar控件将要使用的新调色板的HPALETTE值。

返回值

返回Rebar控件先前调色板的HPALETTE值。

说明

发送这个消息的应用程序将负责删除本消息中所传递的HPALETTE (请参见DeleteObject)。Rebar控件并不会利用这个消息来删除HPALETTE。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版

本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_GETPALETTE。

RB_SETPARENT

设置Rebar控件的父窗口。

RB_SETPARENT

```
wParam = (WPARAM)(HWND) hwndParent;  
lParam = 0;
```

参数

hwndParent: 将要被设置的父窗口句柄。

返回值

返回先前父窗口的句柄; 或者, 如果先前没有父窗口, 则返回NULL。

说明

Rebar控件将利用这个消息向所指定的父窗口发送通告消息。事实上, 这个消息并不会改变Rebar控件的父窗口。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_SETTEXTCOLOR

设置Rebar控件的缺省文本颜色。

RB_SETTEXTCOLOR

```
wParam = 0;  
lParam = (LPARAM)(COLORREF) clrText;
```

参数

clrText: 表示新缺省文本颜色的COLORREF值。

返回值

返回代表先前缺省文本颜色的COLORREF值。

说明

Rebar控件的缺省文本颜色将用来在Rebar控件中绘制文本并且在那些发送此消息之后所添加的区段中绘制文本。当某个区段被添加，或者REBARBANDINFO数据结构中的fMask数据成员被设置了RBBIM_COLORS标志位并且这个数据结构中的clrBack数据成员被修改之后，某个特定区段缺省文本颜色将被覆盖。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

RB_GETTEXTCOLOR。

RB_SETTOOLTIPS

将某个工具提示控件与Rebar控件相关联。

RB_SETTOOLTIPS

```
wParam = (WPARAM) (HWND) hwndToolTip;
lParam = 0;
```

参数

hwndToolTip: 将要被设置的工具提示控件句柄。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RB_SETUNICODEFORMAT

为Rebar控件设置UNICODE字符格式标志。这个消息允许在运行时改变被控件所使用的字符集, 而不必重新创建控件。

```
RB_SETUNICODEFORMAT
    wParam = (WPARAM)(BOOL) fUnicode;
    lParam = 0;
```

参数

fUnicode: 这个参数用来决定由控件所使用的字符集。如果这个参数值为非零, 那么控件将使用UNICODE字符; 否则, 如果这个值为0, 那么控件将使用ANSI字符。

返回值

返回控件的先前UNICODE字符的格式标志。

说明

关于这个消息的更多讨论, 请参见CCM_SETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

RB_GETUNICODEFORMAT。

RB_SHOWBAND

显示或隐藏Rebar控件中某个给定的区段。

```
RB_SHOWBAND
    wParam = (WPARAM)(INT) iBand;
    lParam = (LPARAM)(BOOL) fShow;
```

参数

iBand: Rebar控件中区段基于0的索引。

fShow: 用来说明区段应该显示还是隐藏的布尔值。如果这个值为非零, 那么区段将被显示; 否则, 区段将被隐藏。

返回值

如果操作成功, 则返回非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RB_SIZETORECT

对给定的矩形, 试图找出最佳的区段布局方式。

RB_SIZETORECT

wParam = 0;

lParam = (LPARAM)(LPRECT) prc;

参数

prc: RECT数据结构的地址, 这个数据结构用来说明Rebar控件将要被确定大小的矩形。

返回值

如果发生了布局的改变, 则返回非零值; 否则, 返回0。

说明

Rebar区段将按照适合所给矩形的需要进行重新排列。

带有RBBS_VARIABLEHEIGHT样式的区段将会改变大小, 从而尽可能地与矩形相匹配。

根据新的布局需求, 水平Rebar的高度或竖直Rebar的宽度也可以改变。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

22.4.2 Rebar控件通告消息

NM_CUSTOMDRAW (rebar)

这个通告消息由Rebar控件发送, 用来告知Rebar控件的父窗口所做的绘制操作。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

NM_CUSTOMDRAW

lpNMCustomDraw = (LPNM_CUSTOMDRAW) lParam;

参数

lpNMCustomDraw: NMCUSTOMDRAW数据结构的地址, 在这个数据结构中包含有关于绘制操作的信息。这个数据结构的dwItemSpec数据成员中包含有正在被绘制的区段的标识符。而这个数据结构的ItemlParam数据成员中则包含有正在被绘制的区段的lParam参数。

返回值

应用程序所能够返回的值取决于当前绘制阶段。与这个通告消息相关联的NMCUSTOMDRAW数据结构dwDrawStage数据成员中包含有用来指明绘制阶段的信息。必须返回下列值之一：

如果dwDrawStage等于CDDS_PREPAINT：

CDRF_DODEFAULT：表示控件将绘制自身。在这个绘制阶段，控件不会发送任何NM_CUSTOMDRAW消息。

CDRF_NOTIFYITEMDRAW：表示控件将会告知父窗口任何与项目相关的绘制操作。在进行项目的绘制之前与之后，控件都将会发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYITEMERASE：表示控件将在某个项目被擦除时告知父窗口。控件将在擦除项目之前与之后发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYPOSTERASE：表示控件将在项目被擦除之后告知父窗口。

CDRF_NOTIFYPOSTPAINT：表示控件将在项目被绘制之后告知父窗口。

CDRF_NOTIFYSUBITEMDRAW：在4.71版本中有效，表示控件将在正绘制某个列表视图子项目时告知父窗口。

如果dwDrawStage等于CDDS_ITEMPREPAINT：

CDRF_NEWFONT：表示应用程序为项目指定了新的字体，并且控件将使用新字体。

CDRF_SKIPDEFAULT：表示应用程序将自行绘制项目，Rebar控件不绘制这个项目。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

8.3节。

NM_NCHITTEST (rebar)

当Rebar控件接收到一个WM_NCHITTEST消息时，将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

NM_NCHITTEST

lpnmmouse = (LPNM_MOUSE) lParam;

参数

lpnmmouse：NM_MOUSE数据结构的地址，在这个数据结构中包含有关于本通告消息的信息。在dwItemSpec数据成员中包含有区段索引，在这个区段索引上发生了鼠标击中测试消息；而在pt数据成员中包含有鼠标击中测试消息的鼠标坐标信息。

返回值

如果需要允许Rebar控件执行缺省的鼠标击中测试消息处理, 则返回0; 如果需要覆盖缺省的鼠标击中测试操作, 则应该返回在WM_NCHITTEST中所定义的一个HT*值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RELEASEDCAPTURE (rebar)

这个通告消息用来告知Rebar控件的父窗口, 控件正在释放鼠标捕获。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBN_AUTOSIZE

当带有RBS_AUTOSIZE样式的Rebar控件自动地改变自己的大小时, 控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_AUTOSIZE
    lpnmas = (LPNMRBAUTOSIZE) lParam;
```

参数

lpnmas: NMRBAUTOSIZE数据结构的地址, 在这个数据结构中包含有关于改变大小操作的

信息。

返回值

这个通告消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RBN_BEGINDRAG

当用户开始拖动某个区段时, Rebar控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_BEGINDRAG
    lpnmrb = (LPNMREBAR) lParam;
```

参数

lpnmrb: NMREBAR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

如果需要允许Rebar控件继续执行拖动操作, 则返回0; 否则, 返回非零值用来结束拖动操作。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RBN_CHEVRONPUSHED

当某个人字形被按下时, Rebar控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_CHEVRONPUSHED
    lpnm = (NMREBARCHEVRON) lParam;
```

参数

lpnm: 指向区段中NMREBARCHEVRON数据结构的指针。

返回值

这个通告消息的返回值没有被使用。

说明

当应用程序接收到这个通告消息时, 它将负责显示一个弹出菜单, 在这个菜单中包含有每个隐藏工具的项目。可以使用NMREBARCHEVRON数据结构的rc数据成员来确定弹出菜单的位置。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBN_CHILDSIZE

当某个区段的子窗口发生了大小改变时, Rebar控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

RBN_CHILDSIZE

lprbcs = (LPMREBARCHILDSIZE) lParam;

参数

lprbcs: NMREBARCHILDSIZE数据结构的地址, 在这个数据结构中包含有关于本通告消息信息。

返回值

这个通告消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBN_DELETEDBAND

在某个区段被删除之后, Rebar控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_DELETEDBAND  
lpmrb = (LPNMREBAR) lParam;
```

参数

lpmrb: NMREBAR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

这个通告消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBN_DELETINGBAND

当某个区段将要被删除时, Rebar控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_DELETINGBAND  
lpmrb = (LPNMREBAR) lParam;
```

参数

lpmrb: NMREBAR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

这个通告消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBN_ENDDRAG

当用户结束了对某个区段的拖动操作时, Rebar控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_ENDDRAG
```



```
lpmrb = (LPNMREBAR) lParam;
```

参数

lpmrb: NMREBAR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

这个通告消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RBN_GETOBJECT

当某个对象被拖动到带有RBS_REGISTERDROP样式的Rebar控件中某个区段之上时, 这个控件将发送此通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_GETOBJECT
```

```
lpmmon = (LPNMOBJECTNOTIFY) lParam;
```

参数

lpmmon: NMOBJECTNOTIFY数据结构的地址, 这个数据结构可以包含关于对象被拖动到其上的区段信息, 也可以接收对这个通告消息进行响应的接收消息应用程序所提供的的数据。

返回值

这个通告消息的返回值必须为0。

说明

如果需要接收RBN_GETOBJECT通告消息, 应该调用GLEInitialize或CoInitialize来初始化OLE。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBN_HEIGHTCHANGE

当Rebar控件的高度发生改变时, 控件将发送这个通告消息。这个通告消息是以WM_

NOTIFY消息的形式进行发送的。

```
RBN_HEIGHTCHANGE  
    lpnmhdr = (LPNMHDR) lParam;
```

参数

lpnmhdr: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

这个通告消息的返回值没有被使用。

说明

使用CCS_VERT样式的Rebar控件将在它们的宽度发生改变时发送这个通告消息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

RBN_LAYOUTCHANGED

当用户改变Rebar控件区段的布局时, 控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
RBN_LAYOUTCHANGED  
    lpnmhdr = (LPNMHDR) lParam;
```

参数

lpnmhdr: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

这个通告消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

22.4.3 Rebar控件数据结构

NMRBAUTOSIZE

在这个数据结构中包含有处理RBN_AUTOSIZE通告消息时所使用的信息。

```
typedef struct tagNMRBAUTOSIZE {
    NMHDR hdr;
    BOOL fChanged;
    RECT rcTarget;
    RECT rcActual;
}NMRBAUTOSIZE, *LPNMRBAUTOSIZE;
```

成员

hdr: 包含有关于本通告消息的更多信息的NMHDR数据结构。

fChanged: 这个数据成员用来说明Rebar控件的大小或布局是否发生了改变（如果这个值数据成员非零，那么表示已经发生了改变；否则如果为0，表示没有发生改变）。

rcTarget: 这个数据成员是一个RECT数据结构，在这个数据结构中包含有用来说明Rebar控件正在试图改变自身大小区域所在的矩形。

rcActual: 这个数据成员是一个RECT数据结构，在这个数据结构中包含有Rebar控件自身实际所占用的矩形。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98: 需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMREBAR

在这个数据结构中包含有处理各种Rebar通告消息时所用的信息。

```
typedef struct tagNMREBAR {
    NMHDR hdr;
    DWORD dwMask;
    UINT uBand;
    UINT fStyle;
    UINT wID;
    LPARAM lParam;
}NMREBAR, *LPNMREBAR;
```

成员

hdr: 这个数据成员是一个NMHDR数据结构，在这个数据结构中包含有关于本通告消息的

更多信息。

dwMask: 用来说明数据结构中哪些数据成员包含有合法信息的标志位集合。这个数据成员可以是以下一个或多个值的组合:

RBNM_ID 表示在wID数据成员中包含有合法信息。
RBNM_LPARAM 表示在lParam数据成员中包含有合法信息。
RBNM_STYLE 表示在fStyle数据成员中包含有合法信息。

uBand: 被通告消息所影响的区段基于0的索引。如果没有任何区段受影响, 那么这个参数的值为-1。

fStyle: 区段所使用的样式。这个数据成员是在REBARBANDINFO数据结构fStyle数据成员中详细说明了的一个或多个RBBS_样式。只有在dwMask数据成员中包含有RBNM_STYLE标志位时, 本数据成员才有效。

wID: 由应用程序所定义的区段标识符。只有在dwMask数据成员中包含有RBNM_ID标志位时, 本数据成员才有效。

lParam: 这个数据成员是一个由应用程序所定义的与区段相关联的32位值。只有在dwMask数据成员中包含有RBNM_LPARAM标志位时, 本数据成员才有效。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

NMREBARCHEVRON

在这个数据结构中包含有处理RBN_CHEVRONPUSHED通告消息时所使用的信息。

```
typedef struct tagNMREBARCHEVRON{
    NMHDR hdr;
    UINT uBand;
    UINT wID;
    LPARAM lParam;
    RECT rc;
    LPARAM lParamNM;
}NMREBARCHEVRON, *LPNMREBARCHEVRON;
```

成员

hdr: 这个数据成员是一个NMHDR数据结构, 在这个数据结构中包含有关于本通告消息通告消息的更多信息。

uBand: 正在发送通告消息的区段的索引。

wID: 应用程序为区段所定义的标识符。

lParam: 应用程序为区段所定义的与这个区段相关联的32位值。

rc: 这个数据成员是一个RECT数据结构, 此数据结构用来定义人字形所覆盖的区域。

lParamNM: 这个数据成员是一个由应用程序所定义的32位值。如果在对RB_PUSHCHEVRON消息进行响应时导致了RBN_CHEVRONPUSHED通告消息的发送, 那么这个数据成员将包含此消息的lAppValue值。否则, 这个数据成员将被设置为0。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMREBARCHILDSize

在这个数据结构中包含有用来处理RBN_CHILDSize通告消息时所使用的信息。

```
typedef struct tagNMREBARCHILDSize{
    NMHDR hdr;
    UINT uBand;
    UINT wID;
    RECT rcChild;
    RECT rcBand;
}NMREBARCHILDSize, *LPNMREBARCHILDSize;
```

成员

hdr: 这个数据成员是一个NMHDR数据结构, 此数据结构中包含有关于本通告消息的更多信息。

uBand: 被这个通告消息所影响的区段基于0的索引。如果没有任何区段受影响, 那么这个数据成员将为-1。

wID: 应用程序为区段所定义的标识符。

rcChild: 这个数据成员是一个RECT数据结构, 此数据结构中包含有子窗口的的新大小。在发送用来改变子窗口位置与大小的通告消息时, 这个数据成员也可能会改变。

rcBand: 这个数据成员是一个RECT数据结构, 此数据结构中包含有区段的新大小。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

RBHITTESTINFO

在这个数据结构中包含有鼠标击中操作所特有的信息, 它必须与RB_HITTEST消息一同使用。

```
typedef struct_RB_HITTESTINFO {
    POINT pt;
    UINT flags;
    int iBand;
}RBHITTESTINFO, FAR *LPRBHITTESTINFO;
```

成员

pt: 这个数据成员是一个POINT数据结构, 此数据结构用来描述将要进行鼠标击中测试的点的坐标 (以客户区坐标的形式给出)。

flags: 这个数据成员将接受一个标志值, 用来说明Rebar区段的组件位于由pt数据成员所指定的点。这个数据成员可以是下列值之一:

- RBHT_CAPTION** 表示这个点位于Rebar区段的标题上。
- RBHT_CHEVRON** 表示这个点位于Rebar区段的人字形上 (在5.80版本及其后续版本中有效)。
- RBHT_CLIENT** 表示这个点位于Rebar区段的客户区。
- RBHT_GRABBER** 表示这个点位于Rebar区段夹头条上。
- RBHT_NOWHERE** 表示这个点没有在Rebar区段的任何位置。

iBand: 这个数据成员用来接收由pt所指向的点在Rebar区段上的索引, 并且这个索引是区段基于0的索引。如果没有任何区段在这个鼠标击中测试点上, 那么这个数据成员值为-1。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

REBARBANDINFO

在这个数据结构中包含有用来定义Rebar控件中某个区段的信息。

```
typedef struct tagREBARBANDINFO{
    UINT                cbSize;
```

```

UINT         fMask;
UINT         fStyle;
COLORREF     clrFore;
COLORREF     clrBack;
LPTSTR       lpText;
UINT         cch;
int          iImage;
HWND         hwndChild;
UINT         cxMinChild;
UINT         cyMinChild;
UINT         cx;
HBITMAP      hbmBack;
UINT         wID;
#ifdef _WIN32_IE >= 0x0400
UINT         cyChild;
UINT         cyMaxChild;
UINT         cyIntegral;
UINT         cxIdeal;
LPARAM       lParam;
UINT         cxHeader;
#endif
)REBARBANDINFO, FAR *LPREBARBANDINFO;

```

成员

cbSize: 用来说明数据结构的大小（单位为字节）。在发送任何使用此数据结构的地址作为参数的消息之前，应用程序必须填充这个数据成员。

fMask: 这个数据成员用来说明此数据结构中的哪些数据成员是有效的或者是必须被填充的；这个值可以是下列值的组合：

标志位	描述
RBBIM_BACKGROUND	表示hbmBack数据成员合法或者必须被填充
RBBIM_CHILD	表示hwndChild数据成员合法或者必须被填充
RBBIM_CHILDSIZE	表示cxMinChild、cyMinChild、cyChild、cyMaxChild与cyIntegral数据成员合法或者必须被填充
RBBIM_COLORS	表示clrFore与clrBack数据成员合法或者必须被填充
RBBIM_HEADERSIZE	在4.71版本中有效，表示cxHeader数据成员合法或者必须被填充
RBBIM_IDEALSIZE	在4.71版本中有效，表示cxIdeal数据成员合法或者必须被填充
RBBIM_ID	表示wID数据成员合法或者必须被填充
RBBIM_IMAGE	表示iImage数据成员合法或者必须被填充
RBBIM_LPARAM	在4.71版本中有效，表示lParam数据成员合法或者必须被填充
RBBIM_SIZE	表示cx数据成员合法或者必须被填充
RBBIM_STYLE	表示fStyle数据成员合法或者必须被填充
RBBIM_TEXT	表示lpText数据成员合法或者必须被填充

fStyle: 这个数据成员用来指定区段样式，它可以是以下值的组合：

标 志 位	描 述
RBBS_BREAK	表示区段在新行上
RBBS_CHILDEDGE	表示区段在子窗口的顶部与底部有边
RBBS_FIXEDBMP	表示背景位图在区段大小发生改变时不会移动
RBBS_FIXEDSIZE	表示区段不能改变大小。在这种样式下, 区段上不会显示用来控制大小的夹具条
RBBS_GRIPPERALWAYS	在4.71版本中有效, 表示区段总有用来改变大小的夹具条, 即使是在Rebar控件中只有一个区段
RBBS_HIDDEN	表示区段将不可见
RBBS_NOGRIPPER	在4.71版本中有效, 表示区段将不会出现用来改变大小的夹具条, 即使是在Rebar控件中有多个区段
RBBS_USECHEVRON	在5.80版本中有效, 表示如果区段小于cxIdeal, 那么将显示人字形按钮
RBBS_VARIABLEHEIGHT	在4.70版本有效, 表示区段可以被Rebar控件改变大小。其中, cyIntegral与cyMaxChild数据成员将影响Rebar控件如何改变区段的大小

clrFore和clrBack: 分别表示区段前景色与背景色。如果hbmBack数据成员中指定了背景色位图, 那么这些数据成员将被忽略。缺省情况下, 区段将使用由RB_SETBKCOLOR消息所设置的Rebar控件背景色。如果在上述数据成员中指定了背景色, 那么将使用所指定的背景色。

lpText: 包含将为区段显示文本的缓冲器地址。如果区段信息正在被控件所请求并且fMask数据成员中指定了RBBIM_TEXT, 那么这个数据成员将被初始化为将要接受这个文本的缓冲器地址。

cch: lpText所在缓冲器的大小, 单位为字节。如果控件没有请求这个消息, 那么此数据成员将被忽略。

iImage: 应该被区段首先显示的是图像基于0的索引, 图像列表将由RB_SETBARINFO消息所设置。

hwndChild: 包含在区段中子窗口(若存在)的句柄。

cxMinChild: 子窗口的最小宽度, 单位为像素。区段的宽度不能小于这个值。

cyMinChild: 子窗口的最小高度, 单位为像素。区段的高度不能小于这个值。

cx: 区段的长度, 单位为像素。

hbmBack: 用来作为区段背景位图的句柄。

wID: 控件用来为定制绘图通告消息确定区段的UINT值。这个值在以后也可以用作其他用途。

cyChild: 在4.71版本中有效, 表示区段的初始高度, 单位为像素。除非指定了RBBS_VARIABLEHEIGHT样式, 否则这个数据成员将被忽略。

cyMaxChild: 在4.71版本中有效, 表示区段的最大高度, 单位为像素。除非指定了RBBS_VARIABLEHEIGHT样式, 否则这个数据成员将被忽略。

cyIntegral: 在4.71版本中有效, 表示区段进行增长或收缩时的步进量, 单位为像素。如果区段正在改变大小, 那么它将按照由这个参数所指定的步进量进行大小改变。除非指定了RBBS_VARIABLEHEIGHT样式, 否则这个数据成员将被忽略。

cxIdeal: 在4.71版本中有效, 表示区段的理想宽度, 单位为像素。如果区段被最大化到理想宽度(请参见RB_MAXIMIZEBAND), 那么Rebar控件将试图使得区段的宽度为这个数据成员所指定的宽度值。

lParam: 在4.71版本中有效, 用来表示一个由应用程序定义的32位值。

cxHeader: 在4.71版本中有效,用来表示区段图标的大小,单位为像素。区段头标是指位于区段边与子窗口边之间的区域,在这个区域中将显示区段文本与图像(如果它们被指定)。如果这个参数被指定,那么它将覆盖控件为区段所计算的头标大小。

说明

cxMinChild、cyMinChild与**cx**数据成员能够提供与控件的方向维相关的信息。也就是说,对于水平方向的Rebar控件,**cxMinChild**与**cx**数据成员表示水平度量,而**cyMinChild**表示的是竖直度量。如果Rebar控件使用了**CCS_VERT**样式,那么**cxMinChild**与**cx**数据成员表示的是竖直度量,而**cyMinChild**数据成员表示的是水平度量。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件:在**comctl.h**中进行了声明。

REBARINFO

在这个数据结构中包含有用来描述Rebar控件性状的信息。

```
typedef struct tagREBARINFO {
    UINT          cbSize;
    UINT          fMask;
    HIMAGELIST    hIml;
} REBARINFO, FAR *LPREBARINFO;
```

成员

cbSize: 此数据结构的大小,单位为字节。在发送任何使用这个数据结构的地址作为参数的消息之前,应用程序必须填充这个数据成员。

fMask: 用来描述Rebar控件形状的标志位,目前,Rebar控件仅仅支持一种标志位值:

RBIM_IMAGELIST 表示**hIml**数据成员合法或者必须被填充。

hIml: 图像列表的句柄,Rebar控件将使用所指定的图像列表来获取图像。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者,使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者,使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件:在**comctl.h**中进行了声明。

第23章 状态条

状态条 (status bar) 是一个位于父窗口底部的水平窗口, 应用程序可以在这个水平窗口中显示各式各样的状态信息。为了显示多种类型的信息, 可以将状态条划分为不同的部分。图23-1演示了Microsoft Windows Paint应用程序中的状态条。状态条是位于窗口底部包含Help文本与坐标信息的矩形条。

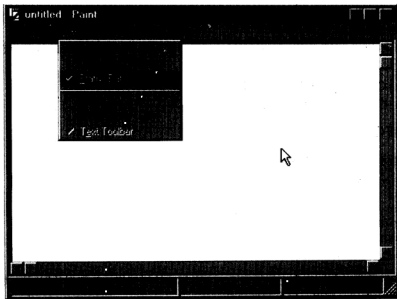


图23-1 状态条

23.1 使用状态条

可以调用 `CreateStatusWindow` 函数, 或者调用 `CreateWindowEx` 函数并指定 `STATUSCLASSNAME` 窗口类, 来创建状态条。为了确信用控件动态链接库 (DLL) 被加载, 首先应该使用 `InitCommonControls` 函数。在创建状态条之后, 就可以将这个状态条划分为几个不同的部分、为每个部分设置文本、使用状态条消息来控制状态条窗口的外观。

23.1.1 状态条类型与样式

状态条的缺省位置是在父窗口的底部, 但可以为状态条指定 `CCS_TOP` 样式, 从而使得状态

条出现在父窗口客户区的顶部。

可以为状态条指定SBARS_SIZEGRIP样式，从而在状态条的右端包含用来改变大小的夹具条。

注意 并不推荐将CCS_TOP样式与SBARS_SIZEGRIP样式组合使用，因为这将会导致用来改变大小的夹具条不能实现其正常的功能。

23.1.2 状态条大小与高度

状态条的窗口处理程序能够自动地设置这个窗口的初始大小与位置，而忽略CreateWindowEx函数中所指定的值。状态条宽度将与父窗口客户区的宽度相同，而它的高度则根据当前为状态条所选中的设备上下文字体以及窗口边界的宽度来共同决定。

当状态条接收到WM_SIZE消息时，窗口处理程序将自动地调整状态条大小。典型情况下，当父窗口的大小发生改变时，父窗口将向状态条发送WM_SIZE消息。

应用程序可以向状态条窗口发送SB_SETMINHEIGHT消息，用来指定状态条的最小高度（单位为像素），从而设置状态条绘制区域的最小高度。其中，状态条的绘制区域并不包括状态条的窗口边界。在自绘状态条中，最小高度信息对于绘制状态条非常有用。关于更多信息，请参见本章后面的23.1.5节“自绘状态条”。

如果向状态条窗口发送SB_GETBORDERS消息，就能够获取状态条的边界宽度。在这个消息中，包含用来接收边界宽度信息的由三个元素组成的数组的地址。

23.1.3 多个部分组成的状态条

一个状态条可以有多个不同的部分，每个部分分别显示不同的文本行。可以向状态条窗口发送SB_SETPARTS消息，并且指定需要创建的状态条部分的数目以及一个整型数组的地址，就可以将状态条划分为多个部分。其中，数组中包含有每个部分所对应的一个元素，而这个元素用来指定每个部分右边界的客户区坐标。

状态条最多可以有256个部分，当然，应用程序通常只使用很少数目的状态条部分。如果向状态条窗口发送SB_SETPARTS消息，就可以获取状态条中所具有的部分数目以及每个部分右边界的坐标。

23.1.4 状态条文本操作

通过发送SB_SETTEXT消息，指定某部分基于0的索引、在这个部分中将要绘制的字符串地址以及绘制字符串所采用的技术，就可以为状态条的任何部分设置文本。其中，绘制字符串文本所采用的技术用来决定文本是否带有边界，以及如果带有边界，那么边界样式是什么。此外，还决定父窗口是否负责文本的绘制工作。关于更多信息，请参见下面将要讨论的23.1.5节“自绘状态条”。

缺省情况下，文本在状态条的指定部分中是按照左对齐方式进行显示的。可以在文本中加入制表符(\t)，从而使文本居中或按照右对齐的方式进行显示。出现在单个制表符右边的文本

将被居中显示，而出现在连续两个制表符右边的文本将按照右对齐的方式显示。

如果需要从状态条中获取文本，则应该使用SB_GETTEXTLENGTH消息与SB_GETTEXT消息。

如果应用程序使用了仅仅拥有一个部分的状态条，那么就可以使用WM_SETTEXT、WM_GETTEXT与WM_GETTEXTLENGTH消息来执行文本操作。这些消息只能对索引号为0的那个部分进行处理，它们将状态条当作静态的文本控件来使用。

如果需要在不创建状态条的情况下显示一行状态信息，则应该使用DrawStatusText函数。这个函数使用绘制状态条所用的技术来绘制状态文本，但它不会自动地设置状态信息的大小与位置。在调用这个函数时，必须指定状态信息的大小与位置，同时也必须指定将要在其中绘制状态信息的窗口所使用的设备上下文。

23.1.5 自绘状态条

可以将状态条中特定的部分指定为自绘部分。利用这个技术，能够使设计者对于窗口部分的外观具有更大的控制权。例如，可以显示位图而不仅仅是文本，或者使用不同的字体绘制文本。

为了将窗口部分定义为自绘部分，应该向状态条发送SB_SETTEXT消息，并且指定将要进行自绘的部分以及SBT_OWNERDRAW绘制技术。当SBT_OWNERDRAW被指定时，IParam参数就是一个由应用程序所定义的32位值，应用程序在绘制这个部分时将会使用这个值。例如，可以指定一个字体句柄、一个位图句柄、一个字符串地址等等。

当状态条需要绘制一个自绘部分时，它将向父窗口发送WM_DRAWITEM消息。这个消息的wParam参数是状态条的子窗口标识符，IParam参数是一个DRAWITEMSTRUCT数据结构的地址。对于状态条的一个自绘部分，DRAWITEMSTRUCT中包含有下列信息：

数据成员	描 述
CtlType	这个数据成员没有被定义，因此目前不应该使用
CtlID	状态条子窗口的标识符
itemID	将要被绘制部分基于0的索引
itemAction	这个数据成员没有被定义，因此目前不应该使用
itemState	这个数据成员没有被定义，因此目前不应该使用
hwndItem	状态条的句柄
hDC	状态条设备上下文的句柄
rcItem	将要被绘制窗口部分的坐标，这个坐标是相对于状态条左上角的坐标
itemData	应用程序所定义的32位值，这个值在SB_SETTEXT消息的IParam参数中指定

23.1.6 简化模式状态条

如果发送SB_SIMPLE消息，那么就将状态条设置为“简化模式”。简化模式的状态条将只显示一个部分。当状态条部分窗口的文本被设置时，这个窗口将被作废，但要等到下一个WM_PAINT消息被发送之后，窗口才进行重绘。利用对消息的等待，可以减少窗口被重绘的次数，从而减少了屏幕出现闪烁的情况。当用户在滚动显示菜单时，简化模式状态条对于显示菜单项目的帮助文本非常有用。

状态条在简化模式下所显示的字符串与在非简化模式下所显示字符串分别采用了不同的管理方法。也就是说，可以将窗口设置为简化模式，为窗口设置文本，同时又可以窗口切换回非简化模式，而不必改变非简化模式下窗口的文本。

在设置简化模式状态条的文本时，可以指定除了SBT_OWNERDRAW之外的任何文本绘制技术。简化模式状态条并不支持自绘功能。

23.1.7 缺省状态条消息处理

本节将描述由预定义的STATUSCLASSNAME类的窗口处理程序所处理的消息。

消 息	缺省处理动作
WM_CREATE	初始化状态条
WM_DESTROY	释放为状态条所分配的资源
WM_GETFONT	返回状态条绘制文本所用的当前字体的句柄
WM_GETTEXT	将状态条中第一个部分的文本复制到缓冲器中。这个消息将返回一个32位值，用来说明所复制的文件长度（单位为字符数）以及绘制文本所采用的技术
WM_GETTEXTLENGTH	这个消息将返回一个32位值，用来说明状态条中第一个部分的文本长度（单位字符数）以及用来绘制文本所采用的技术
WM_NCHITTEST	如果鼠标正在用来改变大小的夹具条上，则返回HTBOTTOMRIGHT值，并且导致系统显示用来表示正在改变大小的鼠标形状。如果鼠标没有用在用来改变大小的夹具条上，那么状态条将把这个消息传递给DefWindowProc函数
WM_PAINT	绘制状态条的无效区域。如果wParam参数不是NULL，那么控件将假设这个值是一个HDC，并且利用这个设备上下文进行区域绘制
WM_SETFONT	为状态条的设备上下文选择字体句柄
WM_SETTEXT	利用缺省的绘制操作（被指定为0），将指定的文本复制到状态条的第一个部分中。如果操作成功，则返回TRUE；否则返回FALSE
WM_SIZE	根据父窗口客户区的当前宽度与状态条当前字体的高度，改变状态条的大小

23.1.8 状态条实例

下面的代码演示了如何创建一个带有可以改变大小的夹具条的状态条，并且演示如何根据父窗口客户区的宽度来将状态条窗口划分为四个相等的部分：

```
//DoCreateStatusBar -creates a status bar and divides it into
// the specified number of parts.
//Returns the handle to the status bar.
//hwndParent -parent window for the status bar.
//nStatusID -child window identifier.
//hInst -handle to the application instance.
//nParts -number of parts into which to divide the status bar.
HWND DoCreateStatusBar(HWND hwndParent,int nStatusID,
HINSTANCE hInst,int nParts)
```

```

{
    HWND hwndStatus;
    RECT rcClient;
    HLOCAL hloc;
    LPINT lpParts;
    int i,nWidth;

    //Ensure that the common control DLL is loaded.
    InitCommonControls();

    //Create the status bar.
    hwndStatus =CreateWindowEx(
        0,                                     //no extended styles
        STATUSCLASSNAME,                     //name of status-bar class
        (LPCTSTR)NULL,                      //no text when first created
        SBARS_SIZEGRIP |                    //includes a sizing grip
        WS_CHILD,                             //creates a child window
        0,0,0,0,                             //ignores size and position
        hwndParent,                          //handle to parent window
        (HMENU)nStatusID,                   //child window identifier
        hinst,                               //handle to application instance
        NULL);                              //no window creation data

    //Get the coordinates of the parent window's client area.
    GetClientRect(hwndParent,&rcClient);

    //Allocate an array for holding the right-edge coordinates.
    hloc =LocalAlloc(LHND,sizeof(int)*nParts);
    lpParts =LocalLock(hloc);

    //Calculate the right-edge coordinate for each part,and
    //copy the coordinates to the array.
    nWidth =rcClient.right /nParts;
    for (i =0;i <nParts;i++){
        lpParts [i ]=nWidth;
        nWidth +=nWidth;
    }

    //Tell the status bar to create the window parts.
    SendMessage(hwndStatus,SB_SETPARTS,(WPARAM)nParts,
        (LPARAM)lpParts);

    //Free the array,and return.
    LocalUnlock(hloc);
    LocalFree(hloc);
    return hwndStatus;
}

```

23.2 在Internet Explorer中更新状态条

Microsoft Internet Explorer中的状态条控件支持下列新特性。

1. 支持图标

可以在状态条中显示图标，SB_SETICON消息用来设置图标。

2. 支持工具提示

状态条现在已经支持工具提示。为了打开工具提示功能，必须在创建状态条时设置SBT_TOOLTIPS样式。SB_SETTIPTTEXT消息与SB_GETTIPTTEXT消息分别用来设置与获取工具提示文本。如果某个状态条部分有一个图标而没有任何文本，或者不能在这个部分中显示所有的文本，这个部分的工具提示将被显示出来。简化模式的状态条不支持工具提示。

3. 对简化模式支持的改进

增加了SB_ISSIMPLE消息，用来决定状态条是否处于简化模式下。增加了SBN_SIMPLEMODECHANGE通告消息，用来告知状态条所有者状态条简化模式发生了改变。

4. 支持背景色

增加了SB_SETBKCOLOR消息，从而允许修改状态条的背景色。

23.3 状态条样式

除了支持标准的窗口样式之外，状态条控件还支持下列样式：

SBARS_SIZEGRIP

这个样式表示状态条控件将在状态条的右端包括一个用来改变大小的夹具条。用来改变大小的夹具条与用来改变大小的边界类似，它是一个矩形区域，用户可以单击以这个矩形区域并且拖动它，从而改变父窗口大小。

SBARS_TOOLTIPS

在5.80版本中有效，与 SBT_TOOLTIPS样式完全相同。在5.00版本及其后续版本中应该使用这个标志位。

SBT_TOOLTIPS

在4.71版本中有效，可以使用这个样式来打开工具提示功能。

23.4 状态条参考

23.4.1 状态条函数

CreateStatusWindow

创建状态窗口，这个状态窗口通常用来显示应用程序的状态信息。此窗口通常出现在父窗口的底部，在其中包含有指定的文本。

```
HWND CreateStatusWindow(
    LONG style,
    LPCTSTR lpszText,
    HWND hwndParent,
    UINT wID
);
```

参数

style: 状态窗口的窗口样式。这个参数必须包括WS_CHILD样式, 同时也应该包括WS_VISIBLE样式。

lpszText: 以null结尾的字符串地址, 这个字符串用来指定状态条第一个部分的状态文本。

hwndParent: 父窗口的句柄。

wID: 状态窗口的控件标识符。窗口处理程序使用这个值来标识发送给父窗口的消息。

返回值

如果操作成功, 则返回状态窗口的句柄; 否则, 返回NULL。如果需要获取扩展的出错信息, 请调用GetLastError函数。

说明

CreateStatusWindow函数调用CreateWindow函数来创建状态窗口, CreateStatusWindow函数将把上述参数不做任何修改地传递给CreateWindow函数, 并且将位置、宽度与高度参数设置为缺省值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

DrawStatusText

DrawStatusText函数将按照状态窗口中带有边界的样式绘制指定的文本。

```
void DrawStatusText (
    HDC hdc,
    LPRECT lprc,
    LPCTSTR pszText,
    UINT uFlags
);
```

参数

hdc: 窗口的显示上下文文本句柄。

lprc: RECT数据结构的指针, 在这个数据结构中包含有将要在其中绘制文本的矩形的坐标(客户区坐标)。这个函数将在指定矩形的边界内部绘制文本边界。

pszText: 以null结尾的字符串的指针, 这个字符串指定了将要显示的文本。出现在这个字符串中的制表符将用来决定字符串是左对齐、右对齐还是居中。

uFlags: 进行文本绘制的标志位, 这个参数可以是下列值的组合:

SBT_NOBORDERS 表示不允许在指定文本的周围绘制边界。

SBT_POPOUT 表示需要绘制高亮边界, 从而使得文本突出。

SBT_RTLDREADING 表示由pszText所指向的字符串将按照与父窗口中文本显示的

相反方向被显示出来。

返回值

这个函数没有返回值。

说明

正常的窗口将按照从左到右 (LTR) 的形式显示文本。当然，窗口可以被镜像，从而显示那些按照从右到左 (RTL) 进行读写的语言文本（例如希伯来语或阿拉伯语）。通常，pszText所指向的字符串将与其父窗口一样的方式显示字符串文本，但如果设置了SBT_RTLREADING，那么pszText字符串将按照与其父窗口相反的方向显示文本。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

导入库：comctl32.lib。

MenuHelp

处理WM_MENUSELECT消息与WM_COMMAND消息，然后在状态窗口中显示当前菜单的帮助文本。

```
void MenuHelp (
    UINT uMsg,
    WPARAM wParam,
    LPARAM lParam,
    HMENU hMainMenu,
    HINSTANCE hInst,
    HWND hwndStatus,
    LPUINT lpwIDs
);
```

参数

uMsg：正在被处理的消息，它可以是WM_MENUSELECT或WM_COMMAND。

wParam：uMsg所指定消息的wParam参数。

lParam：uMsg所指定消息的lParam参数。

hMainMenu：应用程序主菜单的句柄。

hInst：包含字符串资源的模块的句柄。

hwndStatus：状态条窗口的句柄。

lpwIDs：一个数组的地址，在这个数组中包含有字符串资源以及菜单句柄。这个函数将搜索数组，从而查找所选择菜单的句柄。如果找到了所需的句柄，则使用相应的资源标识符来加载适当的帮助字符串。

返回值

这个函数没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

23.4.2 状态条消息**SB_GETBORDERS**

获取状态窗口水平边界与竖直边界的当前宽度信息。

SB_GETBORDERS

wParam = 0;

lParam = (LPARAM)(LPINT) aBorders;

参数

aBorders: 具有三个元素的整型数组的地址。其中, 第一个元素用来接收水平边界的宽度信息, 第二个元素用来接收垂直边界宽度信息, 第三个元素用来接收矩形之间的宽度信息。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

说明

边界可以用来决定窗口的外部边界与包含有文本的窗口矩形之间的空间大小。此外, 边界还能够决定矩形之间的空间大小。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

SB_GETICON

获取状态条某个部分的图标。

SB_GETICON

wParam = (WPARAM)(INT) iPart;

lParam = 0;

参数

iPart: 将要被获取其图标的部分基于0的索引。如果这个参数为-1, 那么状态条将被认为是简化模式 (Simple Mode) 的状态条。

返回值

如果操作成功，则返回图标句柄；否则，返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

SB_GETPARTS

获取状态窗口中所存在的部分数目。这个消息也可以用来接收指定状态部分个数的右边界坐标。

```
SB_GETPARTS
    wParam = (WPARAM) nParts;
    lParam = (LPARAM) (LPINT) aRightCoord;
```

参数

nParts: 将要获取其坐标的状态条部分个数。如果这个参数大于窗口中所存在的部分数目, 那么这个消息将接收现有部分的坐标。

aRightCoord: 一个整型数组的地址, 在这个数组中拥有与nParts参数所指定的数目相同的元素数。数组中的每个元素将接收相应部分的右边界客户区坐标。如果某个元素被设置为-1, 那么这个部分的右边界位置就超出了窗口的右边界。如果需要获取当前存在的部分数目, 则应该将这个参数设置为0。

返回值

如果操作成功, 则返回窗口中存在的部分数目; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

SB_GETRECT

获取状态窗口中某个部分的约束矩形。

```
SB_GETRECT
    wParam = (WPARAM) iPart;
    lParam = (LPARAM) (LPRECT) lprc;
```

参数

iPart: 将要获取其约束矩形的部分基于0的索引。

lpSrc: RECT数据结构的地址, 这个数据结构用来接收约束矩形的信息。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

SB_GETTEXT

SB_GETTEXT消息可以接收来自状态窗口指定部分的文本。

SB_GETTEXT

```
wParam = (WPARAM) iPart;  
lParam = (LPARAM) (LPSTR) szText;
```

参数

iPart: 将要从其中获取文本部分的基于0的索引。

szText: 将要接收文本的缓冲器指针, 事实上, 这个参数是一个以null结尾的字符串。

返回值

返回一个包含有两个16位值的32位值。返回值的低位字部分用来指定文本的长度(单位为字符数), 返回值的高位字部分用来指定用于绘制文本的操作类型。绘制文本的操作类型可以是以下类型之一:

0 表示在绘制文本时, 文本的边界将比窗口的正常位置低。

SBT_NOBORDERS 表示在绘制文本时不使用边界。

SBT_POPOUT 表示在绘制文本时, 文本的边界将比窗口的正常位置高。

SBT_RTLEADING 表示在绘制文本时, 文本的方向将与父窗口中的文本方向相反。

如果文件具有SBT_OWNERDRAW绘制类型, 那么这个消息将返回与文本相关的32位值, 而不是返回文本的长度与操作类型。

说明

正常的窗口将按照从左到右(LTR)的形式显示文本。当然, 窗口可以被镜像, 从而显示那些按照从右到左(RTL)进行读写的语言文本(例如希伯来语或阿拉伯语)。如果设置了SBT_RTLEADING, 那么szText字符串将按照与其父窗口相反的方向显示文本。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

SB_SETTEXT。

SB_GETTEXTLENGTH

SB_GETTEXTLENGTH消息将获取来自状态窗口指定部分文本的长度，单位为字符数。

```
SB_GETTEXTLENGTH
    wParam = (WPARAM) iPart;
    lParam = 0;
```

参数

iPart: 将要从中获取文本部分的基于0的索引。

返回值

返回一个包含有两个16位值的32位值。返回值的低位字部分用来指定文本的长度（单位为字符数），返回值的高位字部分用来指定用于绘制文本的操作类型。绘制文本的操作类型可以是以下类型之一：

0	表示在绘制文本时，文本的边界将比窗口的正常位置低。
SBT_NOBORDERS	表示在绘制文本时不使用边界。
SBT_OWNERDRAW	表示文本将由父窗口进行绘制。
SBT_POPOUT	表示在绘制文本时，文本的边界将比窗口的正常位置高。
SBT_RTLREADING	表示在绘制文本时，文本的方向将与父窗口中的文本方向相反。

说明

正常的窗口将按照从左到右（LTR）的形式显示文本。当然，窗口可以被镜像，从而显示那些按照从右到左（RTL）进行读写的语言文本（例如希伯来语或阿拉伯语）。如果设置了SBT_RTLREADING，那么指定的状态窗口文本将按照与其父窗口相反的方向显示。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

SB_GETTIPTTEXT

获取状态条中某个部分的工具提示文本。当然，状态条必须被创建为SBT_TOOLTIPS样式，从而打开了工具提示功能。

```
SB_GETTIPTTEXT
    wParam = MAKEWPARAM(iPart, nSize);
    lParam = (LPARAM)(LPCTSTR) lpszTooltip;
```

参数

iPart: 将要从中接收工具提示文本部分的基于0的索引。

nSize: lpszTooltip所在缓冲器的大小，单位为字符数。

lpSzTooltip: 将要接收工具提示文本的字符缓冲器地址。

返回值

返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

23.3节。

SB_GETUNICODEFORMAT

获取控件的UNICODE字符格式标志。

SB_GETUNICODEFORMAT

wParam = 0;

lParam = 0;

返回值

返回控件的UNICODE字符格式标志。如果这个值为非零, 那么表示控件正在使用UNICODE字符; 如果这个值为0, 那么表示控件使用ANSI字符。

说明

关于这个消息的更多讨论, 请参见CCM_GETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

SB_SETUNICODEFORMAT。

SB_ISSIMPLE

检查状态条控件, 判断状态条控件是否为简化模式。

SB_ISSIMPLE

wParam = 0;

lParam = 0;

返回值

如果状态条控件处于简单模式，返回非零值；否则，返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

SB_SETBKCOLOR

设置状态条的背景色。

```
SB_SETBKCOLOR
    wParam = 0;
    lParam = (LPARAM)(COLORREF) clrBK;
```

参数

clrBK：用来表示新背景色COLORREF值。如果需要使状态条使用缺省背景色，就应该指定CLR_DEFAULT值。

返回值

返回先前背景色，或者如果背景色是缺省颜色，则返回CLR_DEFAULT。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

SB_SETICON

为状态条中的某个部分设置图标。

```
SB_SETICON
    wParam = (WPARAM)(INT) iPart;
    lParam = (LPARAM)(HICON) hIcon;
```

参数

iPart：将要接收图标部分的基于0的索引。如果这个参数为-1，那么状态条将被认为是一个处于简化模式的状态条。

hIcon: 将要被设置的图标句柄。如果这个值为NULL, 则表示图标从状态条部分中删除。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

说明

状态条不会析构图标。使用这个消息应用程序将负责对图标进行跟踪, 并且析构图标。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

SB_SETMINHEIGHT

设置状态窗口绘制区域的最小高度。

SB_SETMINHEIGHT

```
wParam = (WPARAM) minHeight;  
lParam = 0;
```

参数

minHeight: 窗口的最小高度, 单位为像素。

返回值

没有返回值。

说明

窗口的最小高度是指状态窗口竖直边界的wParam参数值与二倍的边界宽度值(单位为像素)之和。为了对窗口进行重绘, 应用程序必须向状态窗口发送WM_SIZE消息。其中, WM_SIZE消息的wParam参数与lParam参数应该被设置为0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

SB_SETPARTS

设置状态窗口中的部分数目并且设置每个部分的右边界坐标。

SB_SETPARTS

```
wParam = (WPARAM) nParts;  
lParam = (LPARAM) (LPINT) aWidths;
```


参数

nParts: 将要被设置部分的数目 (这个参数不能大于 256)。

aWidths: 一个整型数组的指针, 其中, 这个数组中的元素数目是 **nParts** 参数中所指定的数目。数组中的每个元素将指定相应部分的右边界客户区坐标位置。如果某个元素为 -1, 则说明相应部分的右边界超出了窗口的边界范围。

返回值

若操作成功, 则返回 TRUE; 否则, 返回 FALSE。

环境需求

Windows NT/2000: 需要使用 Windows NT 3.51 或更新版本。

Windows 95/98: 需要使用 Windows 95 或更新版本。

Windows CE: 需要使用 Windows CE 1.0 或更新版本。

头文件: 在 `commctrl.h` 中进行了声明。

SB_SETTEXT

SB_SETTEXT 消息将设置状态窗口中指定部分的文本。

SB_SETTEXT

```
wParam = (WPARAM) iPart | uType;
lParam = (LPARAM) (LPCTSTR) szText;
```

参数

iPart: 将要被设置文本的部分基于 0 的索引; 如果这个参数被设置为 **SB_SIMPLEID**, 那么状态窗口将被认为是只有一个部分的简化窗口。

uType: 文本绘制操作的类型, 这个参数可以是下列值之一:

0 表示在绘制文本时, 文本的边界将比窗口的正常位置低。

SBT_NOBORDERS 表示在绘制文本时不使用边界。

SBT_OWNERDRAW 表示文本将由父窗口进行绘制。

SBT_POPOUT 表示在绘制文本时, 文本的边界将比窗口的正常位置高。

SBT_RTLDREADING 表示在绘制文本时, 文本的方向将与父窗口中的文本方向相反。

szText: 指向一个以 null 结尾的字符串的指针, 这个字符串指定了将要被设置的文本。如果 **uType** 参数为 **SBT_OWNERDRAW**, 则表示此参数代表的是 32 位数据。父窗口必须对这个数据进行解释并且在接收到 **WM_DRAWITEM** 消息时重绘文本。每个部分的文本不得超过 127 个字符。

返回值

若操作成功, 则返回 TRUE; 否则, 返回 FALSE。

说明

这个消息将会使得已经发生改变的窗口作废, 从而导致这个窗口在接收到下一个 **WM_PAINT** 消息时对新文本进行重绘。

正常的窗口将按照从左到右 (LTR) 的形式显示文本。当然, 窗口可以被镜像, 从而显示那

些按照从右到左（RTL）进行读写的语言文本（例如希伯来语或阿拉伯语）。如果设置了 SBT_RTLEADING，那么szText字符串将按照与其父窗口相反的方向显示。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

SB_GETTEXT。

SB_SETTIPTXT

为状态条中的某个部分设置工具提示文本。状态条必须用SB_TOOLTIPS样式进行创建，从而打开工具提示功能。

```
SB_SETTIPTXT
wParam = (WPARAM)(INT) iPart;
lParam = (LPARAM)(LPCTSTR) lpszTooltip;
```

参数

iPart：将要接收工具提示文本的部分基于0的索引。

lpszTooltip：包含新工具提示文本的字符缓冲器地址。

返回值

返回值没有被使用。

说明

关于更多信息，请参见23.2节“在Internet Explorer中更新状态条”。

在以下两种情况下，将会显示工具提示文本：

- 当状态条中的相应面板只包含有一个图标时。
- 当由于面板大小不够，使得状态条中的相应面板中的文本被截取时。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

SB_SETUNICODEFORMAT

为控件设置UNICODE字符格式标志。这个消息能够在运行时改变由控件所使用的字符集，

而不必重新创建控件。

```
SB_SETUNICODEFORMAT
    wParam = (WPARAM)(BOOL) fUnicode;
    lParam = 0;
```

参数

fUnicode: 这个参数将决定被控件所使用的字符集。如果参数值为非零, 那么控件将使用 UNICODE 字符; 如果这个参数值为 0, 那么控件将使用 ANSI 字符。

返回值

返回控件所使用的先前 UNICODE 字符标志。

说明

关于这个消息的更多信息, 请参见 CCM_SETUNICODEFORMAT 的说明。

环境需求

需要使用动态链接库 Comctl32.dll 的 4.00 版本或更新版本。

Windows NT/2000: 需要使用 Windows NT 4.0 或更新版本。

Windows 95/98: 需要使用 Windows 95 或更新版本。

Windows CE: 不支持 Windows CE。

头文件: 在 commctrl.h 中进行了声明。

参见

SB_GETUNICODEFORMAT。

SB_SIMPLE

这个消息用来指定状态窗口是显示简化文本, 还是显示由先前的 SB_SETPARTS 消息所设置的所有窗口部分。

```
SB_SIMPLE
    wParam = (WPARAM)(BOOL) fSimple;
    lParam = 0;
```

参数

fSimple: 这个参数用来说明进行显示的类型标志。如果参数值为 TRUE, 则表示窗口将显示简化文本; 否则, 如果参数值为 FALSE, 则表示窗口显示多个部分。

返回值

返回值没有被使用。

说明

如果状态窗口正在从非简化模式改变为简化模式; 或者从简化模式改变为非简化模式, 那么窗口将立刻进行重绘。

环境需求

Windows NT/2000: 需要使用 Windows NT 3.51 或更新版本。

Windows 95/98: 需要使用 Windows 95 或更新版本。

Windows CE: 需要使用 Windows CE 1.0 或更新版本。

头文件：在commctrl.h中进行了声明。

23.4.3 状态条通告消息

NM_CLICK (状态条)

这个通告消息用来告知状态条控件的父窗口，用户在控件中单击了鼠标左键。NM_CLICK通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CLICK  
lparam = (LPNMMOUSE) lParam;
```

参数

lparam: NMMOUSE数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。其中，dwItemSpec数据成员包含有被单击的部分索引。

返回值

如果返回TRUE，则表示鼠标单击动作被处理并且取消了由系统提供的缺省处理动作；如果返回FALSE，则表示对鼠标单击动作进行缺省处理。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

NM_DBLCLK (状态条)

这个通告消息用来告知状态条控件的父窗口，用户在控件内部双击了鼠标左键。NM_DBLCLK通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_DBLCLK  
lparam = (LPNMMOUSE) lParam;
```

参数

lparam: NMMOUSE数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。其中，dwItemSpec数据成员包含有被双击的部分索引。

返回值

如果返回TRUE，则表示鼠标双击动作被处理并且取消了由系统提供的缺省处理动作；如果返回FALSE，则表示对鼠标双击动作进行缺省处理。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

NM_RCLICK (状态条)

这个通告消息用来告知状态条控件的父窗口，用户在控件中单击了鼠标右键。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RCLICK
    lpm = (LPNMMOUSE) lParam;
```

参数

lpm: NMMOUSE数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。其中，dwItemSpec数据成员包含有被单击的部分索引。

返回值

如果返回TRUE，则表示鼠标单击动作被处理并且取消了由系统提供的缺省处理动作；如果返回FALSE，则表示对鼠标单击动作进行缺省处理。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RDBLCLK (状态条)

这个通告消息用来告知状态条控件的父窗口，用户在控件中双击了鼠标右键。NM_RDBLCLK通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RDBLCLK
    lpm = (LPNMMOUSE) lParam;
```

参数

lpm: NMMOUSE数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。其中，dwItemSpec数据成员包含有被双击的部分索引。

返回值

如果返回TRUE，则表示鼠标双击动作被处理并且取消了由系统提供的缺省处理动作；如果返回FALSE，则表示对鼠标双击动作进行缺省处理。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

SBN_SIMPLEMODECHANGE

当由于SB_SIMPLE消息而导致状态条的简化模式发生改变时，状态条控件将发送这个通告

消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
SBN_SIMPLEMODECHANGE  
    lpmh = (NMHDR*) lParam;
```

参数

lpmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的信息。

返回值

返回值被状态条所忽略。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。



第24章 选项卡控件

选项卡控件 (tab control) 就像笔记本中的分隔标志以及文件柜中的卡片一样。使用选项卡控件, 应用程序就可以在相同的窗口或对话框区域中定义多个页面。其中每个页面都包含有某类信息, 或者包含有一组控件, 当用户在选中了相应的选项卡时, 应用程序将把它们显示出来。

24.1 关于选项卡控件

可以调用CreateWindowEx函数, 并且指定 WC_TABCONTROL窗口类, 从而创建选项卡控件。被指定的这个窗口类将在通用控件动态链接库 (DLL) 被加载时进行注册。为了确保通用控件DLL被加载, 应该使用InitCommonControls函数。

如果需要添加选项卡, 或者发生了影响选项卡控件外观与行为的动作时, 就可以向选项卡控件发送消息。每个消息都有相应的宏, 也可以使用这些宏而不用显式地发送消息的动作。不能关闭选项卡控件中某个单独的选项卡。但是, 可以通过关闭相应的页面的方法来关闭属性页中的选项卡控件。

24.1.1 关于选项卡控件样式

在创建选项卡控件时, 通过指定选项卡控件的样式, 就可以应用某种选项卡形状。例如, 可以指定选项卡控件中选项卡的对齐方式与通用外观。

如果为选项卡控件指定TCS_BUTTONS样式, 那么就可以使选项卡在外观上与按钮相同。这种类型选项卡控件中的选项卡应该具有与按钮控件相同的功能, 也就是说, 单击某个选项卡应该会导致某个命令的执行, 而不是显示一个页面。由于在典型情况下, 按钮选项卡控件中的显示区没有被使用, 因此也不用绘制边界。

如果将选项卡控件指定为TCS_FOCUSONBUTTONDOWN样式, 那么就可以使得在某个选项卡被单击时接收到输入焦点。典型情况下, 这个样式将与TCS_BUTTONS样式一同使用。可以通过指定TCS_FOCUSNEVER样式的方法, 来使得在某个选项卡被单击时, 选项卡不会接收到输入焦点。

缺省情况下, 选项卡控件只显示一行选项卡。如果不能一次显示所有的选项卡, 那么选项卡控件将显示为一个增减数控件, 从而使得用户可以滚动地查看其他的选项卡。在必要时, 可以指定TCS_MULTILINE样式, 从而使得选项卡控件显示多行选项卡。在这种样式下, 所有的选项卡都可以一次被显示出来。在每行选项卡中, 都是按照左对齐的方式进行显示的, 除非指定了TCS_RIGHTJUSTIFY样式。在这种情况下, 每个选项卡的宽度都会自增, 从而使得每行选项卡都填充选项卡控件的整个宽度。

选项卡控件还会自动地调整每个选项卡的大小, 从而使得选项卡能够适合自己的图标与标号 (若存在)。如果需要将所有的选项卡设置为相同的宽度, 那么就应该指定TCS_FIXEDWIDTH样式。这时, 选项卡控件将把所有的选项卡设置为最宽的那个选项卡宽度; 或者,

可以利用TCS_SETITEMSIZE消息来指定特定的宽度与高度。在每个选项卡内部,控件都将把图标与标号进行居中显示,并且将图标放置在标号的左边。当然,也可以指定TCS_FORCEICONLEFT样式,从而使得图标位于选项卡的左边,而使标号居中。也可以利用TCS_FORCELABELLEFT样式将图标与标号进行左对齐。但是应该注意,不能将TCS_FIXEDWIDTH样式与TCS_RIGHTJUSTIFY样式一同使用。

利用TCS_OWNERDRAWFIXED样式,可以指定父窗口在控件中绘制选项卡。关于更多信息,请参见24.1.7节“自绘选项卡”。

利用TCS_TOOLTIPS样式,可以指定选项卡控件创建一个工具提示控件。关于更多信息,请参见24.1.8节“选项卡控件工具提示”。

24.1.2 选项卡与选项卡属性

选项卡控件中的每个选项卡都包含有一个图标、一个标号以及一个由应用程序所定义的数据,这些信息在TCITEM数据结构中被说明。可以向选项卡控件中添加选项卡、获取选项卡控件中的选项卡数目,获取并设置选项卡的内容以及删除选项卡。所有的选项卡都是按照它们基于0的索引进行标识的。

如果需要向选项卡控件中添加选项卡,就应该使用TCM_INSERTITEM消息,并且指定项目位置以及TCITEM数据结构的地址。可以利用TCM_GETITEM消息与TCM_SETITEM消息来获取并设置某个现有选项卡的内容。对于每个选项卡,可以为它指定一个图标、一个标号或者两个都指定。也可以为每个选项卡指定一个数据,使这个数据与选项卡相关联。

可以利用TCM_GETITEMCOUNT消息来获取选项卡控件中的当前选项卡数目,利用TCM_DELETEITEM消息来删除某个选项卡,利用TCM_DELETEALLITEMS消息来删除选项卡控件中的所有选项卡。

可以将由应用程序所定义的数据分别与每个选项卡相关联。例如,有时可能会希望保存与每个选项卡相关联的页面的信息。缺省情况下,选项卡控件为每个选项卡分配四个字节,用来存放由应用程序所定义的数据。可以利用TCM_SETITEMEXTRA消息来改变为每个选项卡所分配的字节数目。当然,只有当选项卡控件为空时,才能发送这个消息。

由应用程序所定义的数据是由TCITEM数据结构的iParam数据成员来指定的。如果需要使用多于四个字节的应用程序定义的数据,那么就必须自行定义数据结构,而不使用TCITEM数据结构。就像获取与设置某个选项卡的其他信息一样,可以使用TCM_GETITEM消息与TCM_SETITEM消息来获取并设置由应用程序所定义的数据。

所定义的数据结构的第一个数据成员必须是TCITEMHEADER数据结构,并且其他的数据成员必须指定由应用程序所定义的数据。TCITEMHEADER数据结构与TCITEM数据结构相同,只是前者没有iParam数据成员。所定义的数据结构大小与TCITEMHEADER数据结构的大小之差应该等于每个选项卡所多出的字节数目。

24.1.3 显示区域

选项卡控件的显示区域是指应用程序显示其当前页面的区域。典型情况下,应用程序将创

建一个子窗口或对话框，并且设置窗口的大小与位置来适合显示区域。在给定了选项卡控件的窗口矩形之后，就可以利用TCM_ADJUSTRECT消息来计算显示区域的约束矩形。

有时，显示区域必须是一个特定的大小（例如，必须是一个非模态子对话框的大小）。在给定了显示区域的约束矩形之后，就可以使用TCM_ADJUSTRECT消息来计算选项卡控件的相应窗口矩形。

24.1.4 选择选项卡

当用户选择某个选项卡时，选项卡控件将以WM_NOTIFY消息的形式向其父窗口发送通告消息。其中，在选择发生改变之前将发送TCN_SELCHANGING通告消息，而在选择发生改变之后则发送TCN_SELCHANGE通告消息。

在处理TCN_SELCHANGING通告消息时，可以保存将要消失的页面状态。如果需要阻止选择发生改变，则应该返回TRUE。例如，当发现某个控件中有一个非法的设置时，就可能不希望所在的子对话框被切换出去。

在处理TCN_SELCHANGE通告消息时，可以在显示区中显示将要出现的页面。这样，将有可能需要改变显示在子窗口中的信息。而更多的时候，每个页面都包含有一个子窗口或对话框。这时，应用程序就应该析构或隐藏将要消失的子窗口或对话框，并且创建或显示将要出现的子窗口或对话框，从而实现对通告消息的处理。

可以利用TCM_GETCURSEL消息以及TCM_SETCURSEL消息来获取与设置选项卡的当前选择。

24.1.5 选项卡控件图像列表

每个选项卡都可以有一个图标与其相关联，这个图标由选项卡控件图像列表的索引来指定。当创建选项卡控件时，是没有任何图像列表与其相关联的。应用程序可以使用ImageList_Create函数来创建一个图像列表，然后用TCM_SETIMAGELIST消息将这个图像列表分配给选项卡控件。

就像将图像添加到任何其他图像列表中一样，可以将图像添加到选项卡控件的图像列表中。但是，应用程序应该使用TCM_REMOVEIMAGE消息而不是使用ImageList_Remove函数将图像从图像列表中删除，因为这个消息能够确保每个选项卡仍然与图像相关联。

在析构某个选项卡控件时，并不会析构与选项卡控件相关联的图像列表。因此，必须单独地析构图像列表。这一点对于希望将同一个图像列表分配给多个选项卡控件的情况非常有用。

如果需要获取与某个选项卡控件相关联的当前图像列表的句柄，就应该使用TCM_GETIMAGELIST消息。

24.1.6 选项卡大小与位置

选项卡控件中的每个选项卡都有大小与位置。可以设置选项卡大小、获取选项卡的约束矩形或者判断哪个选项卡在指定的位置上。

对于固定宽度的选项卡控件以及自绘选项卡控件，可以利用TCM_SETITEMSIZE消息设置

自己所需的选项卡宽度与高度值。对于其他选项卡控件，其中的每个选项卡大小都是根据选项卡的图标与标号进行计算的。选项卡控件中包含一个边界以及附加的页边空白的空间在内。可以利用TCM_SETPADDING消息设置页边空白的厚度。

可以利用TCM_GETITEMRECT消息来判断某个选项卡控件的当前约束矩形。利用TCM_HITTEST消息，可以判断哪个选项卡（若存在）在给定的位置上。

对于带有TCS_MULTILINE样式的选项卡控件，可以使用TCM_GETROWCOUNT消息来判断选项卡的当前行数。

24.1.7 自绘选项卡

如果选项卡控件具有TCS_OWNERDRAWFIXED样式，那么父窗口必须处理WM_DRAWITEM消息来绘制选项卡。当某个选项卡需要被绘制时，选项卡控件将发送这个消息。其中的IParam参数用来指明DRAWITEMSTRUCT数据结构的地址，其中包含有选项卡的索引、选项卡的约束矩形以及将要在其中绘制选项卡的设备上下文。

缺省情况下，DRAWITEMSTRUCT数据结构的itemData数据成员中包含有TCITEM数据结构中的IParam数据成员的值。但是，如果改变了为每个选项卡所指定的由应用程序所指定的数据，那么itemData数据成员中将包含所定义数据的地址。可以利用TCM_SETITEMEXTRA消息来改变由应用为每个选项卡所定义的数据值。

如果需要指定选项卡控件中项目的大小，那么父窗口就必须处理WM_MEASUREITEM消息。由于自绘选项卡控件中的所有选项卡都具有相同的大小，因此这个消息将只被处理一次。对于大小可以改变的自绘选项卡，就不能应用任何选项卡控件样式。还可以使用TCM_SETITEMSIZE消息来设置选项卡的宽度与高度。

24.1.8 选项卡控件工具提示

可以利用工具提示控件来为选项卡控件中的每个选项卡提供一个简单的描述。具有TCS_TOOLTIPS样式的选项卡控件将在选项卡控件被创建时创建一个工具提示控件，并且在选项卡控件被析构时析构工具提示控件。也可以单独地创建一个工具提示控件，然后将它分配给选项卡控件。

如果在选项卡控件中使用了工具提示控件，那么选项卡控件的父窗口就必须处理TTN_NEEDTEXT通告消息，从而为每个被请求的选项卡提供描述信息。

如果需要为多个选项卡控件使用相同的工具提示控件，那么就必须自行创建工具提示控件并且利用TCM_SETTOOLTIPS消息将它分配给选项卡控件。利用TCM_GETTOOLTIPS消息，可以获取选项卡控件当前工具提示控件的句柄。如果自行创建了工具提示控件，那么就不应该使用TCS_TOOLTIPS样式。关于工具提示控件的更多信息，请参见第25章“工具提示控件”。

24.1.9 缺省选项卡控件消息处理

本节将讨论由选项卡控件所执行的消息处理问题。关于选项卡控件所特有的消息，将在其他章节中进行讨论。

消 息

所执行的消息处理动作

WM_CAPTURECHANGED	如果选项卡控件自己释放鼠标捕获, 那么将不执行任何动作。如果另一个窗口捕获了鼠标动作, 并且发现一个按钮被按下, 那么这个命令将释放按钮
WM_CREATE	分配并初始化一个内部数据结构。如果指定了TCS_TOOLTIPS样式, 那么控件将创建一个工具提示控件
WM_DESTROY	释放在处理WM_CREATE消息的过程中所分配的资源
WM_GETDLGCODE	返回DLGC_WANTARROWS与DLGC_WANTCHARS值的组合结果
WM_GETFONT	返回标号所使用的字体的句柄
WM_KEYDOWN	处理方向键, 并且在适当的情况下改变选择
WM_KILLFOCUS	将那个拥有焦点的选项卡设置为失效, 从而使得它能够被重绘以反映失去焦点状态
WM_LBUTTONDOWN	将消息转发给工具提示控件(若存在), 并且在用户单击选项卡时改变选择。如果用户单击了某个按钮, 控件将重绘这个按钮从而使这个按钮出现被按下的那种外观并且捕获鼠标动作。如果用户单击了某个选项卡或者按钮并且选项卡控件具有TCS_FOCUSONBUTTONDOWN样式, 那么控件将把焦点设置给自身
WM_LBUTTONUP	如果某个按钮被按下, 则释放鼠标。如果鼠标正在某个按钮之上并且处于按下状态, 那么控件将相应地改变选择并且重绘按钮
WM_MOUSEMOVE	向工具提示控件转发消息(若存在)。如果为选项卡控件指定了TCS_BUTTONS样式, 并且鼠标在按下之后维持按下状态, 那么控件将重绘受影响的按钮, 从而给出按钮被提升或者被陷入的外观
WM_NOTIFY	转发由工具提示控件发送的通告消息
WM_PAINT	在显示区周围绘制边界(除非指定TCS_BUTTONS样式), 并且绘制那些与失效矩形相交的选项卡 对于每个选项卡, 它将绘制选项卡体(或者向父窗口发送WM_DRAWITEM消息), 然后绘制选项卡的边界。如果wParam参数不是NULL, 那么控件将假设这个参数值是一个HDC, 并且利用这个设备上下文绘制选项卡
WM_RBUTTONDOWN	向父窗口发送NM_RCLICK通告消息
WM_SETFOCUS	将拥有焦点的那个选项卡设置为失效, 从而使得这个选项卡被重绘以反映获取了焦点的状态
WM_SETFONT	设置用于显示标号的字体
WM_SETREDRAW	设置一个内部标志位的状态, 这个标志位将决定当向选项卡控件中插入和删除项目、字体发生改变时, 是否重绘选项卡控件
WM_SIZE	重新计算选项卡的位置, 并且在这时有可能使某些选项卡控件的部分失效, 从而强制重绘某些或者全部选项卡

24.2 使用选项卡控件

本节将提供两个使用选项卡控件的例子。其中第一个例子演示如何使用选项卡控件在应用程序主窗口的多个页面文本之间进行切换。第二个例子演示了如何使用选项卡控件在一个对话框中的多个控件的页面之间进行切换。

24.2.1 创建选项卡控件

本节中的例子将演示如何创建选项卡控件，并且在应用程序主窗口的客户区显示所创建的选项卡控件。应用程序将在选项卡控件的显示区显示第三个窗口（静态控件）。选项卡控件的父窗口在处理WM_SIZE消息时，将负责确定选项卡控件以及静态控件的位置与大小。

在这个选项卡控件中共有七个选项卡，其中一周的每天各占用一个选项卡。当用户选择某个选项卡时，应用程序将在静态控件中显示出相应日期的星期名。在这个例子中使用了以下全局变量：

```
//Global variables

HINSTANCE g_hinst;           //handle to application
                               //instance
char g_achTemp [256];        //temporary buffer for
                               //strings
HWND g_hwndMain;             //main application window
HWND g_hwndTab;              //tab control
HWND g_hwndDisplay;          //handle to static control
                               // in tab control's display area
```

下面给出的函数将负责创建选项卡控件并且为一周中的每天添加一个选项卡。其中这些天的星期名被定义为字符串资源，并且以IDS_FIRSTDAY开始连续地进行编号（在应用程序的头文件中进行了定义）。父窗口与选项卡控件都必须有WS_CLIPSIBLINGS窗口样式。应用程序的初始化函数将在创建主窗口之后调用这个函数。

```
//DoCreateTabControl -creates a tab control,sized to fit the
//      specified parent window's client area,and
//      adds some tabs.
//Returns the handle to the tab control.
//hwndParent -parent window (the application's
//      main window).

HWND WINAPI DoCreateTabControl(HWND hwndParent)
{
    RECT rcClient;
    HWND hwndTab;
    TCITEM tie;
    int i;

    //Get the dimensions of the parent window's client
    //area,and create a tab control child window of
    //that size.
    GetClientRect(hwndParent,&rcClient);
    InitCommonControls();
    hwndTab =CreateWindow(
        WC_TABCONTROL,"",
        WS_CHILD |WS_CLIPSIBLINGS |WS_VISIBLE,
```

```

    0,0,rcClient.right,rcClient.bottom,
    hwndParent,NULL,g_hinst,NULL
};
if (hwndTab ==NULL)
    return NULL;

//Add tabs for each day of the week.
tie.mask =TCIF_TEXT |TCIF_IMAGE;
tie.iImage =-1;
tie.pszText =g_achTemp;

for (i =0;i <7;i++){
    LoadString(g_hinst,IDS_FIRSTDAY +i,
        g_achTemp,sizeof(g_achTemp));
    if (TabCtrl_InsertItem(hwndTab,i,&tie)==-1){
        DestroyWindow(hwndTab);
        return NULL;
    }
}
return hwndTab;
}

```

下列函数将创建占用了选项卡控件整个显示区的静态控件。应用程序的初始化函数将在创建主窗口与选项卡控件之后调用这个函数。

```

//DoCreateDisplayWindow -creates a child window
// (a static control)to occupy the tab
// control's display area.
//Returns the handle to the static control.
//hwndParent -parent window (the application's
// main window).

HWND WINAPI DoCreateDisplayWindow(HWND hwndParent)
{
    HWND hwndStatic =CreateWindow("STATIC","",
        WS_CHILD |WS_VISIBLE |WS_BORDER,
        0,0,CW_USEDEFAULT,CW_USEDEFAULT,
        hwndParent,NULL,g_hinst,NULL);

    return hwndStatic;
}

```

下面给出的是应用程序窗口处理程序的相应部分。应用程序处理WM_SIZE消息，从而确定选项卡控件以及静态控件的位置与大小。为了确定静态控件的适当位置与大小，这个例子将向选项卡控件发送TCM_ADJUSTRECT消息（利用TabCtrl_AdjustRect宏发送这个消息）。

当选项卡被选中时，选项卡控件将发送WM_NOTIFY消息，用来指定TCN_SELCHANGE通告消息。应用程序通过设置静态控件的文本来处理这个通告消息。

```

//MainWindowProc -processes the message for the
// main window class.
//The return value depends on the message.
//hwnd -handle to the window.
//uMsg -identifier for the message.
//wParam -message-specific parameter.
//lParam -message-specific parameter.

LRESULT CALLBACK MainWindowProc(
    HWND hwnd,
    UINT uMsg,
    WPARAM wParam,
    LPARAM lParam
)
{
    switch (uMsg){
        case WM_SIZE:{
            HDWP hdp;
            RECT rc;

            //Calculate the display rectangle,
            //assuming the tab control is the size of
            //the client area.
            SetRect(&rc,0,0,
                LOWORD(lParam),HIWORD(lParam));
            TabCtrl_AdjustRect(g_hwndTab,FALSE,&rc);

            //Size the tab control to fit the client
            //area.
            hdp =BeginDeferWindowPos(2);
            DeferWindowPos(hdp,g_hwndTab,NULL,0,0,
                LOWORD(lParam),HIWORD(lParam),
                SWP_NOMOVE |SWP_NOZORDER
            );
            //Position and size the static control to
            //fit the tab control's display area,and
            //make sure the static control is in front
            //of the tab control.
            DeferWindowPos(hdp,
                g_hwndDisplay,HWND_TOP,rc.left,
rc.top,
                rc.right -rc.left,rc.bottom -
rc.top,0
            );
            EndDeferWindowPos(hdp);
        }
        break;

        case WM_NOTIFY:

```

```

switch (HIWORD(wParam)){
    case 0:
        .
        .//menu command processing
        .

        case TCN_SELCHANGE:{
            int iPage =TabCtrl_GetCurSel
(g_hwndTab);

            LoadString(g_hinst,IDS_FIRSTDAY
+iPage,

                g_achTemp,sizeof(g_achTemp));
            SendMessage(g_hwndDisplay,
WM_SETTEXT,0

                (LPARAM)g_achTemp);
        }
        break;
    ;
    break;

    .
    .        //additional message processing
    .

    default:
        return DefWindowProc(hwnd,uMsg,wParam,
lParam);
    }
    return 0;
}

```

24.2.2 创建选项式对话框

本节中的例子将演示如何创建对话框，并且这个对话框必须使用选项卡来提供控件的多个页面。其中，主对话框是一个模态对话框。控件的每个页面是由具有WS_CHILD样式的对话框模板来定义的。当某个选项卡被选中时，将为需要显示出的页面创建一个非模态对话框，同时将要消失的页面对话框被析构。

注意 在许多情况下，使用属性页能够更容易地实现具有多个页面的对话框。关于属性页的更多信息，请参见第21章。

主对话框的模板只是定义了两个按钮控件。在处理WM_INITDIALOG消息时，对话框处理程序将创建一个选项卡控件并且为每个子对话框加载对话框模板资源。

这些信息被保存在一个名为DLGHDR的应用程序定义的数据结构中，利用SetWindowLong函数，可以使得一个指向此数据结构的指针与对话框窗口相关联。这个数据结构在应用程序的头文件中进行了定义，其定义方式如下：

```
#define C_PAGES 3

typedef struct tag_dlghdr {
    HWND hwndTab;        //tab control
    HWND hwndDisplay;    //current child dialog box
    RECT rcDisplay;      //display rectangle for the tab
                        //control
    DLGTEMPLATE *apRes [C_PAGES];
}DLGHDR;
```

下列函数将为主对话框处理WM_INITDIALOG消息。此函数能够分配DLGHDR数据结构，为子对话框加载对话框模板资源并创建选项卡控件。

每个子对话框的大小由DLGTEMPLATE数据结构决定。这个函数将检查每个对话框的大小，并且使用TCM_ADJUSTRECT消息的宏来为选项卡控件计算适当的大小。然后，确定对话框的大小并且相应地确定两个按钮的位置。这个例子将利用TabCtrl_AdjustRect宏来发送TCM_ADJUSTRECT消息。

```
VOID WINAPI OnTabbedDialogInit(HWND hwndDlg)
{
    DLGHDR *pHdr = (DLGHDR *)LocalAlloc(LPTR,
sizeof(DLGHDR));
    DWORD dwDlgBase = GetDialogBaseUnits();
    int cxMargin = LOWORD(dwDlgBase)/4;
    int cyMargin = HIWORD(dwDlgBase)/8;
    TCITEM tie;
    RECT rcTab;
    HWND hwndButton;
    RECT rcButton;
    int i;

    //Save a pointer to the DLGHDR structure.
    SetWindowLong(hwndDlg, GWL_USERDATA, (LONG)pHdr);

    //Create the tab control.
    InitCommonControls();
    pHdr->hwndTab = CreateWindow(
        WC_TABCONTROL, "",
        WS_CHILD | WS_CLIPSIBLINGS | WS_VISIBLE,
        0, 0, 100, 100,
        hwndDlg, NULL, g_hinst, NULL
    );
    if (pHdr->hwndTab == NULL){
        //handle error
    }

    //Add a tab for each of the three child dialog boxes.
    tie.mask = TCIF_TEXT | TCIF_IMAGE;
```



```

tie.iImage = -1;
tie.pszText = "First";
TabCtrl_InsertItem(pHdr->hwndTab, 0, &tie);
tie.pszText = "Second";
TabCtrl_InsertItem(pHdr->hwndTab, 1, &tie);
tie.pszText = "Third";
TabCtrl_InsertItem(pHdr->hwndTab, 2, &tie);

//Lock the resources for the three child dialog boxes.
pHdr->apRes [0] = DoLockDlgRes
(MAKEINTRESOURCE(DLG_FIRST));
pHdr->apRes [1] = DoLockDlgRes
(MAKEINTRESOURCE(DLG_SECOND));
pHdr->apRes [2] = DoLockDlgRes
(MAKEINTRESOURCE(DLG_THIRD));
//Determine the bounding rectangle for all child
//dialog boxes.
SetRectEmpty(&rcTab);
for (i = 0; i < C_PAGES; i++){
    if (pHdr->apRes [i]->cx > rcTab.right)
        rcTab.right = pHdr->apRes [i]->cx;
    if (pHdr->apRes [i]->cy > rcTab.bottom)
        rcTab.bottom = pHdr->apRes [i]->cy;
}
rcTab.right = rcTab.right * LOWORD(dwDlgBase)/4;
rcTab.bottom = rcTab.bottom * HIWORD(dwDlgBase)/8;

//Calculate how large to make the tab control, so
//the display area can accommodate all the child
//dialog boxes.
TabCtrl_AdjustRect(pHdr->hwndTab, TRUE, &rcTab);
OffsetRect(&rcTab, cxMargin - rcTab.left,
           cyMargin - rcTab.top);

//Calculate the display rectangle.
CopyRect(&pHdr->rcDisplay, &rcTab);
TabCtrl_AdjustRect(pHdr->hwndTab, FALSE,
&pHdr->rcDisplay);

//Set the size and position of the tab control,
//buttons, and dialog box.
SetWindowPos(pHdr->hwndTab, NULL, rcTab.left,
rcTab.top,
           rcTab.right - rcTab.left, rcTab.bottom -
rcTab.top,
           SWP_NOZORDER);

//Move the first button below the tab control.

```

```

hwndButton = GetDlgItem(hwndDlg, BTN_CLOSE);
SetWindowPos(hwndButton, NULL,
    rcTab.left, rcTab.bottom + cyMargin, 0, 0,
    SWP_NOSIZE | SWP_NOZORDER);

//Determine the size of the button.
GetWindowRect(hwndButton, &rcButton);
rcButton.right -= rcButton.left;
rcButton.bottom -= rcButton.top;

//Move the second button to the right of the first.
hwndButton = GetDlgItem(hwndDlg, BTN_TEST);
SetWindowPos(hwndButton, NULL,
    rcTab.left + rcButton.right + cxMargin,
    rcTab.bottom + cyMargin, 0, 0,
    SWP_NOSIZE | SWP_NOZORDER);

//Size the dialog box.
SetWindowPos(hwndDlg, NULL, 0, 0,
    rcTab.right + cyMargin +
    2 * GetSystemMetrics(SM_CXDLGFRAME),
    rcTab.bottom + rcButton.bottom + 2 * cyMargin +
    2 * GetSystemMetrics(SM_CYDLGFRAME) +
    GetSystemMetrics(SM_CYCAPTION),
    SWP_NOMOVE | SWP_NOZORDER);

//Simulate selection of the first item.
OnSelChanged(hwndDlg);
}

//DoLockDlgRes -loads and locks a dialog
//template resource.
//Returns the address of the locked resource.
//lpzResName -name of the resource

DLGTEMPLATE *WINAPI DoLockDlgRes(LPCSTR lpzResName)
{
    HRSRC hrsrc = FindResource(NULL, lpzResName,
RT_DIALOG);
    HGLOBAL hglb = LoadResource(g_hinst, hrsrc);
    return (DLGTEMPLATE *) LockResource(hglb);
}

```

下列函数将为主对话框处理TCN_SELCHANGE通告消息。此函数析构将要消失页面的对话框（若存在），然后使用CreateDialogIndirect函数为将要出现的页面创建非模态对话框。

```

//OnSelChanged -processes the TCN_SELCHANGE
// notification.
//hwndDlg -handle to the parent dialog box.

```

```

VOID WINAPI OnSelChanged(HWND hwndDlg)
{
    DLGHDR *pHdr = (DLGHDR *)GetWindowLong(
        hwndDlg, GWL_USERDATA);
    int iSel = TabCtrl_GetCurSel(pHdr->hwndTab);

    //Destroy the current child dialog box, if any.
    if (pHdr->hwndDisplay != NULL)
        DestroyWindow(pHdr->hwndDisplay);
    //Create the new child dialog box.
    pHdr->hwndDisplay = CreateDialogIndirect(g_hinst,
        pHdr->apRes [iSel], hwndDlg, ChildDialogProc);
}

```

下列函数为每个子对话框处理WM_INITDIALOG消息。不能指定由CreateDialogIndirect函数所创建对话框的位置。此函数将使用SetWindowPos函数在控件的显示区中确定子对话框的位置。

```

//OnChildDialogInit -Positions the child dialog
// box to fall within the display area of the
// tab control.

VOID WINAPI OnChildDialogInit(HWND hwndDlg)
{
    HWND hwndParent = GetParent(hwndDlg);
    DLGHDR *pHdr = (DLGHDR *)GetWindowLong(
        hwndParent, GWL_USERDATA);
    SetWindowPos(hwndDlg, HWND_TOP,
        pHdr->rcDisplay.left, pHdr->rcDisplay.top,
        0, 0, SWP_NOSIZE);
}

```

24.3 在Internet Explorer中更新选项卡控件

Microsoft Internet Explorer中的选项卡控件支持下列新特性：

项目状态

目前，选项卡控件的项目能够支持项目状态，从而支持TCM_DESELECTALL消息。此外，TCITEM数据结构也支持项目状态值。关于更多信息，请参见24.6节“选项卡控件项目状态”。

扩充的样式

现在，选项卡控件能够支持一些扩充的样式，从而使得控件具有更强的功能。关于更多信息，请参见24.5节“选项卡控件扩充样式”。

数据结构更名

选项卡控件使用的所有数据结构都已经更名，从而与当前的命名规范保持一致，同时又具有向后兼容性。例如，TC_ITEM数据结构被更名为TCITEM数据结构。

24.4 选项卡控件样式

除了支持当前的选项卡控件样式之外，还支持以下样式：

TCS_BOTTOM	在4.70版本中有效,在这种样式下,选项卡出现在选项卡控件的底部。这个值与TCS_RIGHT相等。
TCS_BUTTONS	在这种样式下,选项卡以按钮的形式出现,并且在显示区周围没有绘制边界。
TCS_FIXEDWIDTH	在这种样式下,所有的选项卡都具有相同的宽度。这个样式不能与TCS_RIGHTJUSTIFY样式一同使用。
TCS_FLATBUTTONS	在4.71版本有效,在这种样式下,被选中的选项卡将显示为陷入到背景中的形式,而其他选项卡则显示为与背景在同一平面上。这个样式只能影响具有TCS_BUTTONS样式的选项卡控件。
TCS_FOCUSNEVER	在这种样式下,选项卡控件在被单击时不接受输入焦点。
TCS_FOCUSONBUTTONDOWN	在这种样式下,选项卡控件在被单击时将接收输入焦点。
TCS_FORCEICONLEFT	在这种样式下,图标将与每个固定宽度的选项卡左边对齐。这个样式只能与TCS_FIXEDWIDTH样式一同使用。
TCS_FORCELABELLEFT	在这种样式下,标号将与每个固定宽度的选项卡左边对齐,也就是说,标号将显示在图标的正右边,而不是居中显示。 这个样式只能与TCS_FIXEDWIDTH样式一同使用,并且这个样式隐含地说明了需要使用TCS_FORCEICONLEFT样式。
TCS_HOTTRACK	在4.70版本中有效,在这种样式下,出现在鼠标指针下面的项目将自动显示为高亮形式。可以调用SystemParametersInfo,来检查是否打开了热跟踪功能。
TCS_MULTILINE	在这种样式下,如果有必要,将显示多行选项卡,从而使得所有的选项卡都可以同时可见。
TCS_MULTISELECT	在4.70版本中有效,在这种样式下,可以在按下CTRL键的同时单击选项卡,从而选中多个选项卡。这个样式必须与TCS_BUTTONS样式一同使用。
TCS_OWNERDRAWFIXED	在这种样式下,父窗口将负责绘制选项卡。
TCS_RAGGEDRIGHT	在这种样式下,选项卡所在的行将不会自行拉伸从而充满整个选项卡控件的宽度。这个样式是缺省的。
TCS_RIGHT	在4.70版本中有效,在这种样式下,选项卡将在TCS_VERTICAL样式的选项卡控件右边竖直地出现。这个值等价于TCS_BOTTOM。

TCS_RIGHTJUSTIFY	在这种样式下, 若有必要, 每个选项卡的宽度将自增, 从而使得每行选项卡能够充满选项卡控件的整个宽度。除非指定了TCS_MULTILINE样式, 否则这个窗口样式将被忽略。
TCS_SCROLLOPPPOSITE	在4.70版本中有效, 在这种样式下, 当某个选项卡被选中时, 那些当前不需要的选项卡将滚动到选项卡控件的另一边。
TCS_SINGLELINE	在这种样式下, 只有一行选项卡被显示。如果需要, 用户可以滚动选项卡以查看更多的选项卡。缺省情况下, 这种样式是有效的。
TCS_TABS	在这种样式下, 选项卡将以标记卡的形式出现, 并且在显示区周围绘制了边界。缺省情况下, 这种样式是有效的。
TCS_TOOLTIPS	在这种样式下, 选项卡控件将具有一个相关联的工具提示控件。
TCS_VERTICAL	在4.70版本中有效, 在这种样式下, 选项卡将出现在选项卡控件的左边, 并且选项卡文本竖直地显示。只有在与TCS_MULTILINE样式一同使用时, 这个样式才有效。如果需要使选项卡出现在选项卡控件的右边, 那么应该使用TCS_RIGHT样式。

说明

在创建选项卡控件之后, 可以改变下列样式:

- TCS_BOTTOM
- TCS_BUTTONS
- TCS_FIXEDWIDTH
- TCS_FLATBUTTONS
- TCS_FORCEICONLEFT
- TCS_FORCELABELLEFT
- TCS_MULTILINE
- TCS_OWNERDRAWFIXED
- TCS_RAGGEDRIGHT
- TCS_RIGHT
- TCS_VERTICAL

24.5 选项卡控件扩充样式

选项卡控件可以支持几种扩充样式。这些样式必须使用TCM_GETEXTENDEDSTYLE消息与TCM_SETEXTENDEDSTYLE消息进行处理, 并且在使用时不能与传递给CreateWindowEx的

扩充窗口样式相混淆。

样式标志值	描 述
TCS_EX_FLATSEPARATORS	在4.71版本中有效,在这种样式下,选项卡控件将在选项卡项目之间绘制分隔标记。这个扩充样式仅仅影响那些具有TCS_BUTTONS样式与TCS_FLATBUTTONS样式的选项卡控件。缺省情况下,如果利用TCS_FLATBUTTONS样式来创建选项卡控件,将会自动地设置这个扩充样式。如果不需要分隔标记,就应该在创建选项卡控件之后删除此扩充样式
TCS_EX_REGISTERDROP	在4.71版本中有效,在这种样式下,当某个对象被拖动到选项卡控件中的选项卡项目之上时,选项卡控件将产生TCN_GETOBJECT通告消息,请求拖放目标对象。在设置这个样式之前,应用程序必须调用CoInitialize或者OleInitialize

24.6 选项卡控件项目状态

为了支持TCM_DESELECTALL消息,选项卡控件项目现在可以支持项目状态。此外,TCITEM数据结构也支持项目状态值。

项目状态值	描 述
TCIS_BUTTONPRESSED	在4.70版本中有效,这个状态表示选项卡控件项目被选中。只有在设置了TCS_BUTTONS样式时,这个状态才有意义
TCIS_HIGHLIGHTED	在4.71版本中有效,这个状态表示选项卡控件项目处于高亮显示状态,并且正在使用当前高亮颜色绘制选项卡与文本。在使用高亮颜色时,这个颜色必须是一个真正的混写颜色(interpolation),而不是抖动的颜色

24.7 选项卡控件参考

24.7.1 选项卡控件消息

TCM_ADJUSTRECT

在给一个窗口矩形之后,计算选项卡控件的显示区,或者计算与某个给定选择区相对应的窗口矩形。可以显式地发送这个消息,也可以使用TabCtrl_AdjustRect宏来发送这个消息。

```
TCM_ADJUSTRECT
wParam = (WPARAM)(BOOL) fLarger;
lParam = (LPARAM)(LPRECT) prc;
```

参数

fLarger: 将要执行的操作。如果这个参数为TRUE,那么参数prc将指定一个显示矩形并且获取相应的窗口矩形。如果这个参数为FALSE,那么参数prc将指定一个窗口矩形并且获取相应的显示区。

prc: RECT数据结构的地址,这个数据结构用来指定给定的矩形并且获取被计算的矩形。

返回值

没有返回值。

说明

这个消息只能应用于位于顶部的选项卡控件，而不能应用于位于屏幕两边或者底部的选项卡控件。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_DELETEALLITEMS

从选项卡控件中删除所有的项目。可以显式地发送这个消息，也可以使用TabCtrl_DeleteAllItems宏来发送这个消息。

```
TCM_DELETEALLITEMS
```

```
wParam = 0;
```

```
lParam = 0;
```

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_DELETEITEM

从选项卡控件中删除某个项目。可以显式地发送这个消息，也可以使用TabCtrl_DeleteItems宏来发送这个消息。

```
TCM_DELETEITEM
```

```
wParam = (WPARAM)(int) iItem;
```

```
lParam = 0;
```

参数

iItem：将要被删除的项目索引。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_DESELECTALL

在选项卡控件中重新设置项目, 并且清除所有项目的TCIS_BUTTONPRESSED状态。可以显式地发送这个消息, 也可以使用TabCtrl_DeselectAll宏来发送这个消息。

```
TCM_DESELECTALL
wParam = (WPARAM)(DWORD) fExcludeFocus;
lParam = 0;
```

参数

fExcludeFocus: 用来指定将要被重新设置的项目范围。如果这个参数被设置为FALSE, 那么表示所有的选项卡项目都将被重新设置。如果这个参数被设置为TRUE, 那么表示除了当前选中的选项卡项目之外, 其他所有选项卡项目都将被重新设置。

返回值

这个消息返回值没有被使用。

说明

只有在选项卡控件设置了TCS_BUTTONS样式时, 这个消息才有意义。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_GETCURFOCUS

返回选项卡控件中拥有焦点的项目索引。可以显式地发送这个消息, 也可以使用TabCtrl_GetCurFocus宏来发送这个消息。

```
TCM_GETCURFOCUS
wParam = 0;
lParam = 0;
```

返回值

返回拥有焦点的选项卡项目的索引。

说明

拥有焦点的项目可能与被选中的项目不同。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_GETCURSEL

判断选项卡控件中当前被选中的选项卡。可以显式地发送这个消息, 也可以使用TabCtrl_GetCurSel宏来发送这个消息。

```
TCM_GETCURSEL  
wParam = 0;  
lParam = 0;
```

返回值

如果操作成功, 则返回被选中选项卡的索引; 否则, 如果没有任何选项卡被选中, 则返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_GETEXTENDEDSTYLE

获取选项卡控件当前正在使用的扩充样式。可以显式地发送这个消息, 也可以使用TabCtrl_GetExtendedStyle宏来发送这个消息。

```
TCM_GETEXTENDEDSTYLE  
wParam = 0;  
lParam = 0;
```

返回值

返回一个用来表示选项卡控件当前正在使用的扩充样式的DWORD值, 这个DWORD值是选项卡控件扩充样式的组合。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_GETIMAGELIST

获取与选项卡控件相关联的图像列表。可以显式地发送这个消息, 也可以使用TabCtrl_

GetImageList宏来发送这个消息。

```
TCM_GETIMAGELIST
    wParam = 0;
    lParam = 0;
```

返回值

如果操作成功，则返回图像列表的句柄；否则，返回NULL。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_GETITEM

获取选项卡控件中关于某个选项卡的信息。可以显式地发送这个消息，也可以使用TabCtrl_GetItem宏来发送这个消息。

```
TCM_GETITEM
    wParam = (WPARAM)(int) iItem;
    lParam = (LPARAM)(LPTCITEM) pItem;
```

参数

iItem：选项卡的索引。

pItem：TCITEM数据结构的地址，这个数据结构指明了将要获取的信息并且可以接收关于选项卡的信息。当这个消息被发送时，mask数据成员将指明哪些属性应该被返回。

如果mask数据成员指定了TCIF_TEXT值，那么pszText数据成员值必须包含有用来接收项目文本的缓冲器地址，而且cchTextMax数据成员必须指明缓冲器的大小。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

如果在TCITEM数据结构的mask数据成员中设置了TCIF_TEXT标志位，那么控件将有可能改变这个数据结构的pszText数据成员，使其指向新的文本而不是利用所请求的文本填充缓冲器。控件可能会将pszText数据成员设置为NULL，表明没有任何文本与项目相关联。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_GETITEMCOUNT

获取选项卡控件中选项卡的数目。可以显式地发送这个消息，也可以使用TabCtrl_

GetItemCount宏来发送这个消息。

```
TCM_GETITEMCOUNT
wParam = 0;
lParam = 0;
```

返回值

如果操作成功，则返回选项卡的数目；否则，返回0。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_GETITEMRECT

获取选项卡控件中某个选项卡的约束矩形。可以显式地发送这个消息，也可以使用TabCtrl_GetItemRect宏来发送这个消息。

```
TCM_GETITEMRECT
wParam = (WPARAM)(int) iItem;
lParam = (LPARAM)(RECT FAR *) prc;
```

参数

iItem：选项卡的索引。

prc：RECT数据结构的地址，这个数据结构将接收选项卡的约束矩形。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_GETROWCOUNT

获取选项卡控件中选项卡所占用的行数。可以显式地发送这个消息，也可以使用TabCtrl_GetRowCount宏来发送这个消息。

```
TCM_GETROWCOUNT
wParam = 0;
lParam = 0;
```

返回值

返回选项卡所占用的行数。

说明

只有被设置为TCS_MULTILINE样式的选项卡控件才可能有多行选项卡。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_GETTOOLTIPS

获取与某个选项的控件相关联的工具提示控件的句柄。可以显式地发送这个消息, 也可以使用TabCtrl_GetToolTips宏来发送这个消息。

TCM_GETTOOLTIPS

wParam = 0;

lParam = 0;

返回值

如果操作成功, 则返回工具提示控件的句柄; 否则, 返回NULL。

说明

如果选项卡控件拥有TCS_TOOLTIPS样式, 那么它将创建工具提示控件。也可以利用TCM_SETTOOLTIPS消息, 将一个工具提示控件分配给选项卡控件。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TCM_GETUNICODEFORMAT

获取控件的UNICODE字符格式标志。可以显式地发送这个消息, 也可以使用TabCtrl_GetUnicodeFormat宏来发送这个消息。

TCM_GETUNICODEFORMAT

wParam = 0;

lParam = 0;

返回值

返回控件的UNICODE格式标志。如果返回值为非零, 那么表示控件正在使用UNICODE字符; 如果返回值为0, 那么表示控件正在使用ANSI字符。

说明

关于这个消息的更多讨论, 请参见CCM_GETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TCM_SETUNICODEFORMAT。

TCM_HIGHLIGHTITEM

设置选项卡项目的高亮状态。可以显式地发送这个消息, 也可以使用TabCtrl_HighlightItem宏来发送这个消息。

```
TCM_HIGHLIGHTITEM
    wParam = (WPARAM) idItem;
    lParam = (LPARAM) MAKELONG(fHighlight, 0);
```

参数

idItem: 选项卡控件项目基于0的索引。

fHighlight: 这个参数用来指明将要被设置的高亮状态。如果这个值为TRUE, 那么表示选项卡被设置为高亮状态; 否则, 如果这个值为FALSE, 那么表示选项卡被设置为缺省状态。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_HITTEST

判断哪个选项卡在指定的屏幕位置上 (若存在)。可以显式地发送这个消息, 也可以使用TabCtrl_HitTest宏来发送这个消息。

```
TCM_HITTEST
    wParam = 0;
    lParam = (LPARAM) (LPTCHITTESTINFO) pinfo;
```

参数

pinfo: TCHITTESTINFO数据结构的地址, 这个数据结构用来指定将要被测试的屏幕位置。

返回值

返回选项卡的索引, 如果没有任何选项卡在指定的位置上, 则返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_INSERTITEM

在选项卡控件中插入新的选项卡。可以显式地发送这个消息, 也可以使用TabCtrl_InsertItem宏来发送这个消息。

```
TCM_INSERTITEM
    wParam = (WPARAM)(int) iItem;
    lParam = (LPARAM)(const LPTCITEM) pItem;
```

参数

iItem: 新选项卡的索引。

pItem: TCITEM数据结构的地址, 这个数据结构用来指定选项卡的属性。此外, 这个消息将忽略dwState数据成员与dwStateMask数据成员。

返回值

如果操作成功, 则返回新选项卡的索引; 否则, 返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_REMOVEIMAGE

从选项卡控件的图像列表中删除一个图像。可以显式地发送这个消息, 也可以使用TabCtrl_RemoveImage宏来发送这个消息。

```
TCM_REMOVEIMAGE
    wParam = (WPARAM)(int) iImage;
    lParam = 0;
```

参数

iImage: 将要被删除的图像索引。

返回值

没有返回值。

说明

选项卡控件将更新每个选项卡的图像索引, 因此每个选项卡仍然像以前一样与自己的图像相关联。如果某个选项卡正在使用将要被删除的图像, 那么这个选项卡将被设置为没有图像与之相关联。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_SETCURFOCUS

为选项卡控件中指定的选项卡设置焦点。可以显式地发送这个消息, 也可以使用TabCtrl_SetCurFocus宏来发送这个消息。

```
TCM_SETCURFOCUS
    wParam = (WPARAM)(int) iItem;
    lParam = 0;
```

参数

iItem: 将要获取焦点的选项卡索引。

返回值

没有返回值。

说明

如果选项卡控件拥有TCS_BUTTONS样式(按钮模式), 那么拥有焦点的选项卡将可能与被选择的选项卡不同。例如, 当某个选项卡被选中时, 用户可能会按下箭头键将焦点设置给另一个选项卡, 而不改变被选择的选项卡。在按钮模式下, TCM_SETCURFOCUS将把输入焦点设置给与指定的选项卡相关联的按钮, 但它没有改变被选择的选项卡。

如果选项卡控件没有被指定TCS_BUTTONS样式, 那么改变焦点的同时也将会改变被选择的选项卡。这种情况下, 选择的控件将向父窗口发送TCN_SELCHANGING通告消息与TCN_SELCHANGE通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TCM_GETCURFOCUS。

TCM_SETCURSEL

在选项卡控件中选择一个选项卡。可以显式地发送这个消息, 也可以使用TabCtrl_SetCurSel宏来发送这个消息。

```
TCM_SETCURSEL
    wParam = (WPARAM)(int) iItem;
    lParam = 0;
```

参数

iItem: 将要选择的选项卡索引。

返回值

如果操作成功, 则返回先前被选择的选项卡索引; 否则, 返回-1。

说明

当利用这个消息选择了某个选项卡时, 选项卡控件并不发送TCN_SELCHANGING通告消息或TCN_SELCHANGE通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_SETEXTENDEDSTYLE

设置选项卡控件将要使用的扩充样式。可以显式地发送这个消息, 也可以使用TabCtrl_SetExtendedStyle宏来发送这个消息。

```
TCM_SETEXTENDEDSTYLE
    wParam = (WPARAM) dwExMask;
    lParam = (LPARAM) dwExStyle;
```

参数

dwExMask: 用来说明dwExStyle参数中哪个样式将要受到影响的DWORD值。只有dwExMask参数所指定的扩充样式才会发生改变, 而其他样式将维持不变。如果这个参数为0, 那么dwExStyle参数中所有的样式都将会受到影响。

dwExStyle: 用来指明扩充选项卡控件样式的值, 这个值是选项卡控件扩充样式的组合。

返回值

返回包含有先前选项卡扩充样式的DWORD值。

说明

dwExMask参数允许在不必首先获取现有样式的条件下就修改一个或多个扩充样式。例如, 如果将dwExMask参数设置为TCS_EX_FLATSEPARATORS, 并且将dwExStyle参数设置为0, 那么TCS_EX_FLATSEPARATORS样式将被清除, 但所有的其他样式将维持不变。

出于兼容性的考虑, TabCtrl_SetExtendedStyle宏还没有更新为使用dwExMask参数。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_SETIMAGELIST

向选项卡控件分配一个图像列表。可以显式地发送这个消息，也可以使用TabCtrl_SetImageList宏来发送这个消息。

```
TCM_SETIMAGELIST
wParam = 0;
lParam = (LPARAM)(HIMAGELIST) hIm1;
```

参数

hIm1：将要被分配的选项卡控件的图像列表的句柄。

返回值

返回先前图像列表的句柄，如果先前没有指定图像列表，则返回NULL。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_SETITEM

设置选项卡的某些属性或全部属性。可以显式地发送这个消息，也可以使用TabCtrl_SetItem宏来发送这个消息。

```
TCM_SETITEM
wParam = (WPARAM)(int) iItem;
lParam = (LPARAM)(LPTCITEM) pItem;
```

参数

iItem：选项卡项目的索引。

pItem：TCITEM数据结构的地址，在这个数据结构中包含有新项目的属性。mask数据成员指定哪些属性将被设置。

如果mask数据成员指定了LVIF_TEXT值，那么pszText数据成员就必须是一个以null结尾的字符串的地址，而cchTextMax数据成员将被忽略。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_SETITEMEXTRA

设置选项卡控件中为由应用程序定义的数据所预留的每个选项卡字节数。可以显式地发送这个消息，也可以使用TabCtrl_SetItemExtra宏来发送这个消息。

```
TCM_SETITEMEXTRA
    wParam = (WPARAM)(int)cb;
    lParam = 0;
```

参数

cb: 额外增加的字节数。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

缺省情况下，额外增加的字节数为4。如果应用程序改变了这个数目，那么应用程序就不能使用TCITEM数据结构来获取与设置应用程序为选项卡所定义的数据。相反，应用程序必须自行定义一个新的数据结构，而且所定义的数据结构必须包含TCITEMHEADER数据结构以及由应用程序所定义的数据成员。

只有在选项卡控件没有包含任何选项卡时，应用程序才可以改变额外增加的字节数。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCM_SETITEMSIZE

在固定宽度的选项卡控件或自绘选项卡控件中设置选项卡的宽度与高度。可以显式地发送这个消息，也可以使用TabCtrl_SetItemSize宏来发送这个消息。

```
TCM_SETITEMSIZE
    wParam = 0;
    lParam = MAKELPARAM(cx, cy);
```

参数

cx和cy: 新的宽度与高度值，单位为像素。

返回值

返回原来的宽度与高度值。其中，宽度值放置在返回值的低位字部分，高度值放置在返回值的高位字部分。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_SETMINTABWIDTH

设置选项卡控件中项目的最小宽度。可以显式地发送这个消息，也可以使用TabCtrl_SetMinTabWidth宏来发送这个消息。

```
TCM_SETMINTABWIDTH
    wParam = 0;
    lParam = (LPARAM)(INT) cx;
```

参数

cx：将要为选项卡项目所设置的最小宽度值。如果这个参数被设置为-1，那么表示选项卡控件将使用缺省的选项卡宽度。

返回值

返回用来表示先前最小选项卡宽度的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_SETPADDING

设置选项卡控件中每个选项卡图标与标号周围的空白区大小。可以显式地发送这个消息，也可以使用TabCtrl_SetPadding宏来发送这个消息。

```
TCM_SETPADDING
    wParam = 0;
    lParam = MAKELPARAM(cx, cy);
```

参数

cx和cy：分别表示水平空白区大小与竖直空白区大小，单位为像素。

返回值

没有返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TCM_SETTOOLTIPS

为选项卡控件分配一个工具提示控件。可以显式地发送这个消息，也可以使用TabCtrl_SetToolTips宏来发送这个消息。

```
TCM_SETTOOLTIPS
    wParam = (WPARAM)(HWND) hwndTT;
    lParam = 0;
```

参数

hwndTT: 工具提示控件的句柄。

返回值

没有返回值。

说明

可以使用TCM_GETTOOLTIPS消息，来获取与选项卡控件相关联的工具提示控件。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TCM_SETUNICODEFORMAT

为控件设置UNICODE字符格式表示。这个消息允许在运行时改变控件所使用的字符集，而不必重新创建控件。可以显式地发送这个消息，也可以使用TabCtrl_SetUnicodeFormat宏来发送这个消息。

```
TCM_SETUNICODEFORMAT
    wParam = (WPARAM)(BOOL) fUnicode;
    lParam = 0;
```

参数

fUnicode: 用来决定由控件所使用的字符集。如果这个值为非零，那么表示控件将使用UNICODE字符。如果这个值为0，那么表示控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式标志。

说明

关于这个消息的更多讨论，请参见CCM_SETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

TCM_GETUNICODEFORMAT。

24.7.2 选项卡控件宏

TabCtrl_AdjustRect

在给定窗口矩形之后，计算选项卡控件的显示区；或者，计算与指定显示区相对应的窗口矩形。可以使用这个宏，也可以通过显式发送TCM_ADJUSTRECT消息的方法来完成此动作。

```
VOID TabCtrl_AdjustRect (
    HWND hwnd,
    BOOL fLarger,
    RECT FAR *prc
);
```

参数

hwnd：选项卡控件的句柄。

fLarger：将要执行的操作。如果这个参数为TRUE，那么prc参数将指定显示矩形并且接收相应的窗口矩形信息。如果这个参数为FALSE，那么prc参数将指定窗口矩形并接收相应的显示区。

prc：RECT数据结构的地址，这个数据结构将指定一个给定的矩形并且接收被计算的矩形。

返回值

没有返回值。

说明

这个消息只能应用于位于窗口顶部的选项卡控件，而不能应用于位于窗口两边以及窗口底部的选项卡控件。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_DeleteAllItems

从选项卡控件中删除所有的项目。可以使用这个宏，也可以通过显式发送TCM_DELETEALLITEMS消息的方法来完成此动作。

```
BOOL TabCtrl_DeleteAllItems (
    HWND hwnd
);
```

参数

hwnd: 选项卡控件的句柄。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_DeleteItem

从选项卡控件中删除一个项目。可以使用这个宏, 也可以通过显式发送TCM_DELETEITEM消息的方法来完成此动作。

```
BOOL TabCtrl_DeleteItem (
    HWND hwnd,
    int item
);
```

参数

hwnd: 选项卡控件的句柄。

item: 将要被删除的项目的索引。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_DeselectAll

重新设置选项卡控件中的项目, 并且清除选项卡的TCIS_BUTTONPRESSED状态。可以使用这个宏, 也可以通过显式发送TCM_DESELECTALL消息的方法来完成此动作。

```
void TabCtrl_DeselectAll (
    HWND hwndTab,
    UINT fExcludeFocus
);
```

参数

hwndTab: 选项卡控件的句柄。

fExcludeFocus: 这个参数用来指定将要进行重新设置的项目范围。如果这个参数被设置为FALSE, 那么所有的选项卡项目都将被重新设置。如果这个参数被设置为TRUE, 那么除了当前

被选中的选项卡项目之外，其他所有选项卡项目将被重新设置。

返回值

返回值没有被使用。

说明

只有当选项卡控件设置了TCS_BUTTONS样式时，这个消息才有意义。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_GetCurFocus

返回选项卡控件中拥有焦点的项目索引。可以使用这个宏，也可以通过显式发送TCM_GETCURFOCUS消息的方法来完成此动作。

```
int TabCtrl_GetCurFocus (  
    HWND hwnd  
) ,
```

参数

hwnd：选项卡控件的句柄。

返回值

返回拥有焦点的选项卡项目的索引。

说明

拥有焦点的选项卡项目可能与被选中的选项卡项目不同。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_GetCurSel

判断选项卡控件中当前被选中的选项卡。可以使用这个宏，也可以通过显式发送TCM_GETCURSEL消息的方法来完成此动作。

```
int TabCtrl_GetCurSel (  
    HWND hwnd  
) ;
```

参数

hwnd: 选项卡控件的句柄。

返回值

如果操作成功, 则返回被选中的选项卡的索引; 否则, 如果没有任何选项卡被选中, 则返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_GetExtendedStyle

获取选项卡控件当前正在使用的扩充样式。可以使用这个宏, 也可以通过显式发送TCM_GETEXTENDEDSTYLE消息的方法来完成此动作。

```
DWORD TabCtrl_GetExtendedStyle (  
    HWND hwndTab  
) ;
```

参数

hwndTab: 选项卡控件句柄。

返回值

返回用来表示选项卡控件当前正在使用的扩充样式的DWORD值, 这个值是选项卡控件扩充样式的组合。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_GetImageList

获取与某个选项卡控件相关联的图像列表。可以使用这个宏, 也可以通过显式发送TCM_GETIMAGELIST消息的方法来完成此动作。

```
HIMAGELIST TabCtrl_GetImageList (  
    HWND hwnd  
) ;
```


返回值

如果操作成功，则返回图像列表的句柄；否则，返回NULL。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_GetItem

获取选项卡控件中某个选项卡的信息。可以使用这个宏，也可以通过显式发送TCM_GETITEM消息的方法来完成此动作。

```
BOOL TabCtrl_GetItem (
    HWND hwnd,
    int item,
    LPTCITEM pitem
);
```

参数

hwnd：选项卡控件的句柄。

item：选项卡的索引。

pitem：TCITEM数据结构的地址，这个数据结构指定了将要获取的信息并且接收关于选项卡的信息。将发送这个消息时，mask数据成员将指定哪些属性应该被返回。

如果mask数据成员指定了TCIF_TEXT值，那么 pszText数据成员则必须包含将要接收项目文本的缓冲器地址，cchTextMax数据成员必须指定缓冲器的大小。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

如果在TCITEM数据结构的mask数据成员中设置了TCIF_TEXT标志位，那么控件将有可能改变pszText数据成员的值，使其指向新的文本而不是利用被请求的文本来填充缓冲器。控件可能会将pszText数据成员设置为NULL，用来表明没有文本与项目相关联。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_GetItemCount

获取选项卡控件中的选项卡数目。可以使用这个宏，也可以通过显式发送TCM_GETITEMCOUNT

消息的方法来完成此动作。

```
int TabCtrl_GetItemCount (
    HWND hwnd,
);
```

参数

hwnd: 选项卡控件的句柄。

返回值

如果操作成功, 则返回选项卡项目的数目; 否则, 返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_GetItemRect

获取选项卡控件中某个选项卡的约束矩形。可以使用这个宏, 也可以通过显式发送TCM_GETITEMRECT消息的方法来完成此动作。

```
BOOL TabCtrl_GetItemRect (
    HWND hwnd,
    int item,
    RECT FAR *prc
);
```

参数

hwnd: 选项卡控件的句柄。

item: 选项卡的索引。

prc: RECT数据结构的地址, 这个数据结构将接收选项卡的约束矩形。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_GetRowCount

获取选项卡控件中选项卡当前所占用的行数。可以使用这个宏, 也可以通过显式发送TCM_GETROWCOUNT消息的方法来完成此动作。

```
int TabCtrl_GetRowCount (
```

```

    HWND hwnd,
);

```

参数

hwnd: 选项卡控件的句柄。

返回值

返回选项卡所占用的行数。

说明

只有TCS_MULTILINE样式的选项卡控件才可能会有多行选项卡。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_GetToolTips

获取与选项卡控件相关联的工具提示控件的句柄。可以使用这个宏, 也可以通过显式发送TCM_GETTOOLTIPS消息的方法来完成此动作。

```

int TabCtrl_GetToolTips (
    HWND hwnd,
);

```

参数

hwnd: 选项卡控件的句柄。

返回值

如果操作成功, 则返回工具提示控件的句柄; 否则, 返回NULL。

说明

如果选项卡控件具有TCS_TOOLTIPS样式, 那么它将创建工具提示控件。也可以使用TCM_SETTOOLTIPS消息, 向选项卡控件分配工具提示控件。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_GetUnicodeFormat

获取选项卡控件的UNICODE字符格式标志。可以使用这个宏, 也可以通过显式发送TCM_GETUNICODEFORMAT消息的方法来完成此动作。

```

BOOL TabCtrl_GetUnicodeFormat (

```

```
HWND hwnd
);
```

参数

hwnd: 控件的句柄。

返回值

返回控件的UNICODE字符标志。如果这个值为非零, 那么表示控件正在使用UNICODE字符; 如果这个值为0, 那么表示控件正在使用ANSI字符。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TabCtrl_SetUnicodeFormat。

TabCtrl_HighlightItem

设置选项卡项目的高亮状态。可以使用这个宏, 也可以通过显式发送TCM_HIGHLIGHTITEM消息的方法来完成此动作。

```
BOOL TabCtrl_HighlightItem (
    HWND hwndTab,
    INT idItem,
    WORD fHighlight
);
```

参数

hwndTab: 选项卡控件的句柄。

idItem: 选项卡控件项目基于0的索引。

fHighlight: 这个参数用来指定将要被设置的高亮状态。如果这个值为非零, 那么表示选项卡处于高亮状态。如果这个值为0, 那么表示选项卡被设置为其缺省状态。

返回值

若操作成功, 则返回一个非零值; 否则, 返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件：在commctrl.h中进行了声明。

TabCtrl_HitTest

判断哪个选项卡在指定的屏幕位置（若存在）。可以使用这个宏，也可以通过显式发送TCM_HITTEST消息的方法来完成此动作。

```
int TabCtrl_HitTest (
    HWND hwnd,
    LPTCHITTESTINFO pinfo
);
```

参数

hwnd：选项卡控件的句柄。

pinfo：TCHITTESTINFO数据结构的地址，这个数据结构指定了将要被测试的屏幕位置。

返回值

返回选项卡的索引，如果没有任何选项卡位于指定的位置上，则返回-1。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_InsertItem

在选项卡控件中插入一个新选项卡。可以使用这个宏，也可以通过显式发送TCM_INSERTITEM消息的方法来完成此动作。

```
int TabCtrl_InsertItem (
    HWND hwnd,
    int iItem,
    const LPTCITEM pitem
);
```

参数

hwnd：选项卡控件的句柄。

iItem：新选项卡的索引。

pitem：TCITEM数据结构的地址，这个数据结构用来指定选项卡的属性。本消息将忽略dwState数据成员与dwStateMask数据成员。

返回值

如果操作成功，则返回新选项卡的索引；否则，返回-1。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_RemoveImage

从选项卡控件的图像列表中删除一个图像。可以使用这个宏, 也可以通过显式发送TCM_REMOVEIMAGE消息的方法来完成此动作。

```
void TabCtrl_RemoveImage (  
    HWND hwnd,  
    int iImage  
) ,
```

参数

hwnd: 选项卡控件的句柄。

iImage: 将要被删除的图像的索引。

返回值

没有返回值。

说明

选项卡控件将更新每个选项卡的图像索引, 从而使得每个选项卡都能够像从前一样与自己的图像相关联。如果某个选项卡正在使用将要被删除的图像, 那么这个选项卡将被设置为没有图像与之相关联。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetCurFocus

为选项卡控件中指定的选项卡设置焦点。可以使用这个宏, 也可以通过显式发送TCM_SETCURFOCUS消息的方法来完成此动作。

```
VOID TabCtrl_SetCurFocus (  
    HWND hwnd,  
    int iItem  
) ,
```

参数

hwnd: 选项卡控件的句柄。

iItem: 将要获取焦点的选项卡基于0的索引。

返回值

没有返回值。

说明

如果选项卡控件拥有TCS_BUTTONS样式(按钮模式),那么拥有焦点的选项卡将可能与被选择的选项卡不同。例如,当某个选项卡被选中时,用户可能会按下箭头键将焦点设置给另一个选项卡,而不改变被选择的选项卡。在按钮模式下,TabCtrl_SetCurFocus宏将把输入焦点设置给与指定的选项卡相关联的按钮,但它没有改变被选择的选项卡。

如果选项卡控件没有被指定TCS_BUTTONS样式,那么改变焦点的同时也将会改变被选择的选项卡。这种情况下,选择了控件将向父窗口发送TCN_SELCHANGING通告消息与TCN_SELCHANGE通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TabCtrl_GetCurFocus, TCM_GETCURFOCUS。

TabCtrl_SetCurSel

在选项卡控件中选择选项卡。可以使用这个宏,也可以通过显式发送TCM_SETCURSEL消息的方法来完成此动作。

```
int TabCtrl_SetCurSel (
    HWND hwnd,
    int iItem
);
```

参数

hwnd: 选项卡控件的句柄。

iItem: 将要被选择的选项卡的索引。

返回值

如果操作成功,则返回先前被选中的选项卡索引;否则,返回-1。

说明

当使用TCM_SETCURSEL消息选中了一个选项卡时,选项卡控件并不发送TCN_SELCHANGING通告消息与TCN_SELCHANGE通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetExtendedStyle

设置选项卡控件将要使用的扩充样式。可以使用这个宏,也可以通过显式发送TCM_

SETEXTENDEDSTYLE消息的方法来完成此动作。

```
DWORD TabCtrl_SetExtendedStyle (
    HWND hwnd, Tab
    DWORD dwExStyle
);
```

参数

hwndTab: 选项卡控件的句柄。

dwExStyle: 包含新选项卡扩充样式的值, 这个值是选项卡控件扩充样式 (extended styles) 的组合。

返回值

返回包含有先前选项卡扩充样式的DWORD值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetImageList

为选项卡控件分配一个图像列表。可以使用这个宏, 也可以通过发送TCM_SETIMAGELIST消息的方法来完成此动作。

```
BOOL TabCtrl_SetImageList (
    HWND hwnd,
    HIMAGELIST himl
);
```

参数

hwnd: 选项卡控件的句柄。

himl: 将要被分配的选项卡控件的图像列表的句柄。

返回值

返回先前图像列表的句柄, 如果先前不存在图像列表, 只返回NULL。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetItem

设置选项卡的某些属性或全部属性。可以使用这个宏，也可以通过显式发送TCM_SETITEM消息的方法来完成此动作。

```
BOOL TabCtrl_SetItem (
    HWND hwnd,
    int iItem,
    LPTCITEM pItem
);
```

参数

hwnd: 选项卡控件的句柄。

iItem: 项目的索引。

pItem: TCITEM数据结构的地址，这个数据结构中包含有新项目属性。mask数据成员将指定哪些属性被设置。

如果mask数据成员指定了LVIF_TEXT值，那么pszText数据成员就是一个以null结尾的字符串的地址，而cchTextMax数据成员将被忽略。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetItemExtra

设置选项卡控件中为由应用程序所定义的数据所预留的每个选项卡字节数。可以使用这个宏，也可以通过显式发送TCM_SETITEMEXTRA消息的方法来完成此动作。

```
BOOL TabCtrl_SetItemExtra (
    HWND hwnd,
    int cb
);
```

参数

hwnd: 选项卡控件的句柄。

cb: 额外增加的字节数。

返回值

若操作成功，则返回TRUE；否则，返回FALSE。

说明

缺省情况下，额外增加的字节数为4。如果应用程序改变了这个数目，那么应用程序就不能使用TCITEM数据结构来获取与设置应用程序为选项卡所定义的数据。相反，应用程序必须自行

定义一个新的数据结构，而且所定义的数据结构必须包含TCITEMHEADER数据结构以及由应用程序所定义的数据成员。

只有在选项卡控件没有包含任何选项卡时，应用程序才可以改变额外增加的字节数。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_SetItemSize

在固定宽度的选项卡控件或自绘选项卡控件中设置选项卡的宽度高度。可以使用这个宏，也可以通过显式发送TCM_SETITEMSIZE消息的方法来完成此动作。

```
DWORD TabCtrl_SetItemSize (  
    HWND hwnd,  
    int cx,  
    int cy  
);
```

参数

hwnd：选项卡控件的句柄。

cx和cy：新的宽度与高度值，单位为像素。

返回值

返回原来的宽度与高度值。其中，宽度值放置在返回值的低位字部分，高度值放置在返回值的高位字部分。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TabCtrl_SetMinTabWidth

设置选项卡控件中项目的最小宽度。可以使用这个宏，也可以通过显式发送TCM_SETMINTABWIDTH消息的方法来完成此动作。

```
int TabCtrl_SetMinTabWidth (  
    HWND hwndTab,  
    INT cx  
);
```

参数

hwndTab：选项卡控件的句柄。

cx: 将要为选项卡项目所设置的最小宽度值。如果这个参数被设置为-1, 那么表示选项卡控件将使用缺省的选项卡宽度。

返回值

返回用来表示先前最小选项卡宽度的INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetPadding

设置选项卡控件中每个选项卡图标与标号周围的空白区大小。可以使用这个宏, 也可以通过显式发送TCM_SETPADDING消息的方法来完成此动作。

```
void TabCtrl_SetPadding (
    HWND hwnd,
    int cx,
    int cy
);
```

参数

hwnd: 选项卡控件的句柄。

cx和**cy**: 分别表示水平空白区大小与垂直空白区大小, 单位为像素。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetToolTips

为选项卡控件分配一个工具提示控件。可以使用这个宏, 也可以通过显式发送TCM_SETTOOLTIPS消息的方法来完成此动作。

```
void TabCtrl_SetToolTips (
    HWND hwndTab,
```

```
HWND hwndTT  
);
```

参数

hwndTab: 选项卡控件的句柄。

hwndTT: 工具提示控件的句柄。

返回值

没有返回值。

说明

可以使用TCM_GETTOOLTIPS消息, 来获取与选项卡控件相关联的工具提示控件。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TabCtrl_SetUnicodeFormat

为控件设置UNICODE字符格式标志。这个消息允许在运行时改变控件所使用的字符集, 而不必重新创建控件。可以使用这个宏, 也可以通过显式发送TCM_SETUNICODEFORMAT消息的方法来完成此动作。

```
BOOL TabCtrl_SetUnicodeFormat (  
    HWND hwnd,  
    BOOL fUnicode  
);
```

参数

hwnd: 选项卡控件的句柄。

fUnicode: 用来决定由控件所使用的字符集。如果这个值为非零, 那么表示控件将使用UNICODE字符。如果这个值为0, 那么表示控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式标志。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TabCtrl_GetUnicodeFormat。

24.7.3 选项卡控件通告消息

NM_CLICK (选项卡)

这个通告消息用来告知选项卡控件的父窗口，用户在控件上单击了鼠标左键。NM_CLICK 通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CLICK  
lpmh = (LPNMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

返回值被选项卡控件忽略。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RCLICK (选项卡)

这个通告消息用来告知选项卡控件的父窗口，用户在控件上单击了鼠标右键。NM_RCLICK 通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RCLICK  
lpmh = (LPNMHDR) lParam;
```

参数

lpmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

返回值被选项卡控件忽略。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NM_RELEASEDCAPTURE (选项卡)

这个通告消息用来告知选项卡控件父窗口，控件正在释放鼠标捕获。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE  
lpmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址, 在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TCN_FOCUSCHANGE

这个通告消息用来告知选项卡控件的父窗口, 按钮的焦点发生了改变。

TCN_FOCUSCHANGE

参数

没有任何参数。

返回值

没有返回值。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

TCN_GETOBJECT

当选项卡控件拥有TCS_EX_REGISTERDROP扩充样式, 并且某个对象被拖动到控件中的一个选项卡项目之上时, 选项卡控件将发送这个通告消息。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

TCN_GETOBJECT

lpnmn = (LPNMOBJECTNOTIFY) lParam;

参数

lpnmon; NMOBJECTNOTIFY数据结构的地址。在这个数据结构中包含有关于选项卡项目以及被拖动的对象的信息;同时,这个数据结构还接收应用程序对这个消息进行响应时返回的数据。

返回值

处理这个通告消息的应用程序必须返回0。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者,使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者,使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TCN_KEYDOWN

这个通告消息用来告知选项卡控件的父窗口,某个键被按下。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
TCN_KEYDOWN
    pnm = (NMTCKEYDOWN FAR *) lParam;
```

参数

pnm: NMTCKEYDOWN数据结构的地址。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCN_SELCHANGE

这个通告消息用来告知选项卡控件的父窗口,当前选中的选项卡发生了改变。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
TCN_SELCHANGE
    lpmhldr = (LPNMHDR) lParam;
```

参数

lpmhldr: NMHDR数据结构的地址。其中, dwndFrom数据成员用来存放选项卡控件的句柄。idFrom数据成员用来存放选项卡控件的子窗口标识符。code数据成员的值为TCN_SELCHANGE。

返回值

没有返回值。

说明

如果需要判断当前被选中的选项卡，应该使用TabCtrl_GetCurSel宏。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

TCN_SELCHANGING。

TCN_SELCHANGING

这个通告消息用来告知选项卡控件的父窗口，当前被选中的选项卡将要发生改变。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
TCN_SELCHANGING  
    lpnmhdr = (LPMHDM) lParam;
```

参数

lpnmhdr：NMHDR数据结构的地址。其中，hwndFrom数据成员用来存放选项卡控件的句柄。idFrom数据成员用来存放选项卡控件的子窗口标识符。code数据成员的值是TCN_SELCHANGING。

返回值

如果需要阻止选择发生改变，则返回TRUE；如果允许改变的发生，则返回FALSE。

说明

如果需要判断当前被选中的选项卡，可以使用TabCtrl_GetCurSel宏。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

TCN_SELCHANGE。

24.7.4 选项卡控件数据结构

NMTCKEYDOWN

在这个数据结构中包含有选项卡控件中被按下的键的信息。它将被TCN_KEYDOWN通告消息所使用，正在逐步取代TC_KEYDOWN数据结构。


```
typedef struct tagNMTCKEYDOWN {
    NMHDR hdr;
    WORD wVKey;
    UINT flags;
} NMTCKEYDOWN;
```

成员

hdr: 这个数据成员是NMHDR数据结构, 其中包含有关于本通告消息的信息。

wVKey: 虚拟键代码。

flags: 这个数据成员的值与WM_KEYDOWN消息中lParam参数的值相同。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCHITTESTINFO

这个数据结构中包含有关于鼠标击中测试的信息。此数据结构正在逐步取代TC_HITTESTINFO数据结构。

```
typedef struct tagTCHITTESTINFO {
    POINT pt;
    UINT flags;
} TCHITTESTINFO, FAR *LPTCHITTESTINFO;
```

成员

pt: 将要进行鼠标击中测试的位置 (在客户坐标下)。

flags: 这个数据成员是用来接收鼠标击中测试结果的变量。选项卡控件将把这个数据成员设置为下列值之一:

TCHT_NOWHERE

表示这个位置没有在任何选项卡之上。

TCHT_ONITEM

表示这个位置在某个选项卡之上, 但没有在选项卡的图标或文本之上。对于自绘选项卡控件, 如果指定位置在某个选项卡的任何位置之上, 都将会设置这个值。

TCHT_ONITEMICON

表示这个位置在某个选项卡的图标之上。

TCHT_ONITEMLABEL

表示位置在某个选项卡的文本之上。

TCHT_ONITEM是TCHT_ONITEMICON与TCHT_ONITEMLABEL的位或结果。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TCM_HITTEST。

TCITEM

这个数据成员用来设置或接收某个选项卡项目的属性。它将被TCM_INSERTITEM消息、TCM_GETITEM消息与TCM_SETITEM消息所使用, 此数据结构正在逐步取代TC_ITEM数据结构。

```
typedef struct tagTCITEM {
    UINT mask;
#ifdef (_WIN32_IE >= 0x0300)
    DWORD dwState;
    DWORD dwStateMask;
#else
    UINT lpReserved1;
    UINT lpReserved2;
#endif
    LPCTSTR pszText;
    int cchTextMax;
    int iImage;
    LPARAM lParam;
} TCITEM, FAR *LPTCITEM;
```

成员

mask: 这个数据成员用来指定将要被获取或设置数据成员信息。它可以是下列值的组合:

TCIF_IMAGE	表示iImage数据成员有效。
TCIF_PARAM	表示lParam数据成员有效。
TCIF_RTLREADING	表示由pszText数据成员所指向的字符串将按照父窗口中显示文本的方向相反的方向显示文本。
TCIF_STATE	在4.70版本有效, 表示dwState数据成员有效。
TCIF_TEXT	表示pszText数据成员有效。

dwState: 在4.70版本中有效, 如果正在获取信息, 那么这个数据成员用来指定项目的当前状态。如果在设置项目信息, 这个数据成员中包含有将要为这个项目所设置的状态值。如果需要了解合法的选项卡控件项目状态列表, 请参见24.6节。在TCM_INSERTITEM消息中, 这个数据成员被忽略。

dwStateMask: 在4.70版本中有效, 用来指明dwState数据成员中的哪些位包含有合法信息。在TCM_INSERTITEM消息中, 这个数据成员被忽略。

lpReserved1: 在4.00版本中有效, 这个数据成员目前没有被使用。

lpReserved2: 在4.00版本中有效, 这个数据成员目前没有被使用。

pszText: 以null结尾的字符串地址, 这个字符串中包含有在项目信息被设置时所需的选项卡文本。如果正在获取项目信息, 那么这个数据成员用来指定将要接收选项卡文本的缓冲器地址。

cchTextMax: 由pszText数据成员所指向的缓冲器的大小。如果这个数据结构不是处于接收信息的状态, 那么这个数据成员将忽略。

iImage: 选项卡控件图像列表的索引, 如果选项卡没有图像, 那么这个数据成员的值为-1。

lParam: 与选项卡项目相关联的由应用程序所定义的数据。如果每个选项卡中由应用程序所定义的数据不是占用四个字节, 那么应用程序必须自行定义一个数据结构, 并且使用所定义的数据结构来取代TCITEM数据结构。其中, 由应用程序所定义的数据结构的第一个数据成员必须是TCITEMHEADER数据结构。

说明

正常的情况下, 窗口将按照从左到右(LTR)的方式显示文本。当然, 窗口可以被镜像, 从而显示诸如希伯来语或阿拉伯语之类按照从右到(RTL)方式进行读写的语言。一般来说, pszText将按照与其父窗口相同的方向显示文本。但是, 如果设置了TCIF_RTREADING, 那么pszText将按照与其父窗口相反的方向显示文本。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TCITEMHEADER

这个数据结构用来设置或者接收选项卡的属性, 它将被TCM_INSERTITEM消息、TCM_GETITEM消息与TCM_SETITEM消息所使用。这个数据结构正在逐步取代TC_ITEMHEADER数据结构。

```
typedef struct tagTCITEMHEADER{
    UINT mask;
    UINT lpReserved1;
    UINT lpReserved2;
    LPTSTR pszText;
    int cchTextMax;
    int iImage;
}TCITEMHEADER, FAR *LPTCITEMHEADER;
```

成员

mask: 用来指定哪些数据成员将被接收或被设置的标志位, 它可以是下列值的组合:

TCIF_IMAGE 表示iImage数据成员有效。

TCIF_RTREADING 表示由pszText所指向的字符串将按照与其父窗口相反的方向显

示文本。

TCIF_TEXT 表示pszText数据成员有效。

lpReserved1: 这是一个预留的数据成员, 因此不应该使用它。

lpReserved2: 这是一个预留的数据成员, 因此不应该使用它。

pszText: 以null结尾的字符串地址, 这个字符串中包含有在项目信息被设置时所需的选项卡文本。如果正在获取项目信息, 那么这个数据成员用来指定将要接收选项卡文本的缓冲器地址。

cchTextMax: 由pszText数据成员所指向的缓冲器的大小。如果这个数据结构不是处于接收信息的状态, 那么这个数据成员将忽略。

iImage: 选项卡控件图像列表的索引, 如果选项卡没有图像, 那么这个数据成员的值为-1。

说明

正常情况下, 窗口将按照从左到右 (LTR) 的方式显示文本。当然, 窗口可以被镜像, 从而显示诸如希伯来语或阿拉伯语之类按照从右到 (RTL) 方式进行读写的语言。一般来说, pszText将按照与其父窗口相同的方向显示文本。但是, 如果设置了TCIF_RTLEADING, 那么pszText将按照与其父窗口相反的方向显示文本。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。



第25章 工具提示控件

工具提示控件是一个小的弹出窗口，在这个窗口中显示了用来描述应用程序中某个工具用途的一行文本。这里所指的工具可以是一个窗口（例如一个子窗口、控件），也可以是窗口客户区中由应用程序所定义的矩形区域。

25.1 关于工具提示控件

在大多数时候，工具提示控件是隐藏的，只有当用户将鼠标移动到某个工具上并且鼠标在这个工具上停留了大约半秒钟之后，工具提示控件才会出现。工具提示控件出现在鼠标的旁边，并且在用户单击了鼠标按钮或者将鼠标移出工具时消失。一个工具提示控件可以支持任何数目的工具。图25-1演示了一个标准的工具提示控件，它与工具条控件中的一个按钮相关联。工具提示也可以有多行的样式，在这种样式下会出现多行文本；或者具有球形样式，在这种样式下，工具提示像卡通气球一样显示带有圆角以及茎干的图示。

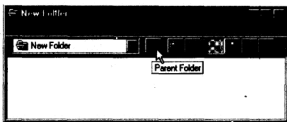


图25-1 工具提示控件

25.1.1 创建工具提示控件

如果需要创建工具提示控件，就应该调用CreateWindowEx函数，并指定TOOLTIPS_CLASS窗口类。所指定的这个窗口类在通用控件动态链接库（DLL）被加载时进行注册，为了确保这个DLL被加载，应该在应用程序中调用InitCommonControls函数。此外，必须明确地声明工具提示控件位于屏幕最前端。否则，工具提示控件就有可能被父窗口所覆盖。下列代码段演示如何创建工具提示控件：

```
HWND hwndTip = CreateWindowEx(NULL, TOOLTIPS_CLASS, NULL,
    WS_POPUP | TTTS_NOPREFIX | TTTS_ALWAYSTIP,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    hwndParent, NULL, hinstMyDll,
```

```

NULL);

SetWindowPos(hwndTip,HWND_TOPMOST,0,0,0,0,
    SWP_NOMOVE | SWP_NOSIZE | SWP_NOACTIVATE);

```

工具提示控件的窗口处理程序将自动设置此控件的大小、位置与可见性。工具提示窗口的高度是根据当前为工具提示控件的设备上下文选择的字体高度所决定的。工具提示窗口宽度则根据工具提示窗口中当前字符串的长度来动态变化。

25.1.2 激活

工具提示控件可以处于活跃状态或非活跃状态。当它处于活跃状态时，如果鼠标出现在某个工具上，工具提示控件将出现。当工具提示控件处于非活跃状态时，即使是鼠标出现在某个工具上，工具提示控件也不会出现。TTM_ACTIVATE消息可用来激活与关闭某个工具提示控件。

25.1.3 工具的类型

工具提示控件可以支持任意数目的工具。为了支持某个特定的工具，必须向工具提示控件发送TTM_ADDTOOL消息，从而在工具提示控件中注册此工具。在所发送的消息中，应该包含有TOOLINFO数据结构的地址，这个数据结构能够向工具提示控件提供显示工具的文本时所需信息。此外，需要使用cbSize数据成员并且指定数据结构的大小。

工具提示控件支持以窗口形式实现的工具（例如子窗口或控件窗口），也支持以窗口中客户区内矩形区域形式所实现的工具。当向工具提示控件中添加一个以矩形区域形式实现的工具时，TOOLINFO的hwnd数据成员必须指定包含有这个区域的窗口句柄，rect数据成员必须指定区域约束矩形的客户坐标。此外，uld数据成员必须指定应用程序为工具所定义的标识符。

当向工具提示控件中添加一个以窗口形式实现的工具时，TOOLINFO的uld数据成员必须包含有工具的窗口句柄。此外，uFlags数据成员必须指定为TTF_IDISHWND值，这个值用来告知工具提示控件将uld数据成员解释为窗口句柄。

25.1.4 工具提示文本

在向工具提示控件中添加一个工具时，TOOLINFO上数据结构的lpstrText数据成员必须指定将要为工具显示的字符串地址。在添加这个工具之后的任何时候，都可以通过发送TTM_UPDATETIPTTEXT消息的方法来改变这个文本。

如果lpstrText数据成员的高位字部分为0，那么这个数据成员的低位字部分必须是字符串资源的标识符。当工具提示控件需要此文本时，系统将从由TOOLINFO数据结构hinst数据成员所定义的应用程序实例中加载指定的字符串资源。

如果lpstrText数据成员中指定了LPSTR_TEXTCALLBACK值，那么工具提示将在需要为工具显示文本时，发送TTM_NEEDTEXT通告消息，告知由TOOLINFO数据结构hwnd数据成员所指定的窗口。在这个通告消息中，包含有TOOLTIPTTEXT数据结构的地址，这个数据结构则包含



有窗口的句柄以及应用程序为工具所定义的标识符。窗口将检查这个数据结构，并且决定需要为其显示文本的工具，并且利用工具提示控件需要显示的字符串信息来填充适当的数据成员。

注意 工具提示文本的最大长度为 80 个字符。关于更多信息，请参见 NMTTDISPINFO 数据结构。

许多应用程序都创建包含有与窗口菜单命令相对应工具的工具条。对于这些工具，工具提示控件可以很方便地显示与菜单项目相同的文本。系统能够自动地从传递给工具提示控件的所有字符串中析取出“与”符号(&)加速字符，除非控件使用了 TTS_NOPREFIX 样式。

可以使用 TTM_GETTEXT 消息获取工具的文本。

25.1.5 将鼠标消息转发到工具提示控件

通常，在鼠标位于某个区域的上方时，工具提示将被显示出来。这里所说的区域，一般是指由某个工具（例如按钮控件）所定义的矩形。但是，窗口只会向包含有鼠标的那个窗口中发送与鼠标相关的消息，而不会直接发送给工具提示控件自身。因此，为了使工具提示控件能够在适当的时机、适当的位置显示工具提示信息，必须能够将与鼠标相关的信息转发给工具提示控件。

在以下条件下，可以自动地实现消息的转发：

- 工具是一个在此工具的 TOOLINFO 数据结构中进行了定义的控件或者是一个矩形。
- 与工具相关联的窗口在工具提示控件所在的线程中。

如果满足这两个条件，那么在利用 TTM_ADDTOOL 向工具提示控件添加工具时，应该设置此工具 TOOLINFO 数据结构 uFlags 数据成员的 TTF_SUBCLASS 标志位。然后，必要的鼠标消息将自动地被转发给工具提示控件。

对于大多数应用情况，设置 TTF_SUBCLASS 来向控件发送鼠标消息就已经足够了。但是，如果工具提示控件与工具窗口之间没有直接的联系，那么这样做并不会得到所期望的结果。例如，如果某个工具是在由应用程序所定义的窗口中作为矩形区域而实现的，那么窗口处理程序将接收到鼠标消息。设置 TTF_SUBCLASS 能够确保鼠标消息被传递给控件，但如果工具是作为由系统所指定的窗口来实现的，那么鼠标消息将被发送给这个窗口并且不能直接被应用程序所使用。在这种情况下，就必须将这个窗口定义为一个子类，或者使用消息转发机制来访问鼠标消息。然后，必须使用 TTM_RELAYEVENT 来显式地将鼠标消息转发给工具提示控件。关于如何使用 RELAYEVENT，请参见 25.2.2 节“在对话框控件中使用工具提示控件”。

当某个工具提示控件接收到 WM_MOUSEMOVE 消息时，此控件将判断鼠标是否在工具的约束矩形上。若是，那么工具提示控件将设置一个定时器。在到达所设置的时间间隔之后，工具提示控件将检查鼠标的位置以判断鼠标是否发生了移动。如果鼠标没有移动，工具提示控件将获取此工具的文本，并且显示工具提示。工具提示控件会一直显示此工具提示窗口，直到接收到一个被转发过来的鼠标按钮弹起消息或鼠标按钮被按下消息，或者接收到一个用来说明鼠标移出了工具所在约束矩形的 WM_MOUSEMOVE 消息时，工具提示窗口才消失。

事实上，每个工具提示控件都有三个与其相关联的过期时间间隔。其中，初始间隔时间是指在工具提示窗口被显示出来之前，鼠标必须在工具的约束矩形中停留的时间长度；重显时间

间隔是指当鼠标从一个工具移动到另一个工具时，后一个工具所对应的工具提示被显示之前所经历的延迟；工具弹出时间间隔是指在工具提示窗口隐藏之前，这个窗口维持显示状态的时间，也就是说，在工具提示窗口被显示之后，如果鼠标仍然在约束矩形只能维持静止不动的状态，工具提示窗口自动隐藏所经历的时间。可以利用TTM_SETDELAYTIME消息来修改这三个过期时间值。

如果某个应用程序包含有作为矩形区域方式所实现的工具，并且控件的大小或位置发生改变，那么应用程序就可以使用TTM_NEWTOOLRECT消息，向工具提示控件报告所发生的改变。对于以窗口形式实现的工具，应用程序并不需要报告大小与位置改变信息，这是因为工具提示控件可以使用工具的窗口句柄来判断鼠标是否在工具之上，而不必使用工具的约束矩形进行判断。

当某个工具提示将要被显示时，工具提示控件将向所有者窗口发送TTN_SHOW通告消息。而当某个工具提示将要被隐藏时，所有者窗口将接收到TTN_POP通告消息。每个通告消息都是以WM_NOTIFY消息的上下文形式进行发送的。

25.1.6 工具提示鼠标击中测试

TTM_HITTEST消息可以用来获取处在某个特定的点上的工具提示控件信息。在这个消息中有一个TTHITTESTINFO数据结构，此数据结构包含有一个窗口句柄、点的坐标以及TOOLINFO数据结构的地址。工具提示控件将判断某个工具是否在给定的点上，如果在给定的点上，则用关于本工具的信息来填充TOOLINFO数据结构。

25.1.7 其他消息

TTM_GETCURRENTTOOL消息与TTM_GETTOOLINFO消息将利用已经被工具提示控件所注册的某个工具的信息来填充TOOLINFO数据结构。TTM_SETTOOLINFO消息可以用来改变工具提示控件为某个特定的点所维护的信息。TTM_DELTOOL消息还可以从工具提示控件中删除某个工具。

25.1.8 缺省工具提示控件消息处理

本节将描述窗口处理程序为TOOLTIPS_CLASS窗口类所处理的消息。

消 息	描 述
WM_CREATE	这个消息用来确信工具提示控件具有WS_EX_TOOLWINDOW窗口样式与WS_POPUP窗口样式。此外，它还能够分配内存与初始化内部变量
WM_DESTROY	这个消息用来释放为工具提示控件所分配的资源
WM_GETFONT	这个消息将返回工具提示控件用来绘制文本的字体的句柄
WM_MOUSEMOVE	这个消息将用来隐藏工具提示窗口
WM_PAINT	这个消息将用来绘制工具提示窗口
WM_SETFONT	这个消息将用于设置工具提示控件用来绘制文本的字体的句柄
WM_TIMER	如果工具的位置发生了改变，或者如果鼠标移到了工具之外，那么这个消息将隐藏工具提示窗口。否则，将显示工具提示窗口
WM_WININCHANGE	这个消息用来重新设置与系统度量相关的内部变量

25.2 使用工具提示控件

本节将提供演示如何创建工具提示控件以及如何使用对话框工具提示控件的例子。

25.2.1 创建工具提示控件

下面的例子将演示如何创建工具提示控件以及如何向其中添加工具。在这个例子中，将首先在窗口的客户区中创建一个矩形格栅，然后使用于TTM_ADDTOOL消息将每个矩形添加到工具提示控件中。为了能够使鼠标消息自动地传递给工具提示控件，这个例子为TOOLINFO数据结构中的uFlags数据成员设置了TTF_SUBCLASS标志位。

```
//DoCreateTooltip -creates a tooltip control and
//adds some tools to it.
//
//Returns the handle of the tooltip control if successful,
//or NULL otherwise.
//
//hwndOwner -handle of the owner window
//
//Global variable
// g_hinst -handle of the application instance
extern HINSTANCE g_hinst;

HWND DoCreateTooltip(HWND hwndOwner)
{
    HWND hwndTT; //handle of the tooltip
    int row,col; //rows and columns
    TOOLINFO ti; //tool information
    int id =0; //offset to string identifiers
    static char *szTips [NUM_TIPS ]= //tooltip text
    {
        "Cut","Copy","Paste","Undo","Open","Save"
    };

    //Ensure that the common control DLL is loaded,
    //and create a tooltip control.
    InitCommonControls();

    hwndTT =CreateWindow(TOOLTIPS_CLASS,(LPSTR)NULL,TTS_ALWAYSTIP,
        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
        NULL,(HMENU)NULL,g_hinst,NULL);

    if (hwndTT ==(HWND)NULL)
        return (HWND)NULL;

    //Divide the client area into a grid of rectangles,
    //and add each rectangle to the tooltip.
```

```

for (row =0;row <MAX_ROWS ;row++){
    for (col =0;col <MAX_COLS;col++){
        ti.cbSize =sizeof(TOOLINFO);
        ti.uFlags =TTF_SUBCLASS;
        ti.hwnd =hwndOwner;
        ti.hinst =g_hinst;
        ti.uId =(UINT)id;
        ti.lpszText =(LPTSTR)szTips [id++];
        ti.rect.left =col *CX_COLUMN;
        ti.rect.top =row *CY_ROW;
        ti.rect.right =ti.rect.left +CX_COLUMN;
        ti.rect.bottom =ti.rect.top +CY_ROW;

        if (!SendMessage(hwndTT,TM_ADDTOOL,0,
            (LPARAM) (LPTOOLINFO)&ti))
            return NULL;
    }
}

return hwndTT;
}

```

25.2.2 在对话框控件中使用工具提示控件

下面的例子中包含一系列由应用程序所定义的函数，这些函数将在对话框中实现工具提示控件。其中，DoCreateDialogTooltip函数将创建一个工具提示控件，并且使用EnumChildWindows函数对对话框中的控件进行枚举。用来枚举的处理程序EnumChildProc函数将在工具提示控件中对每个控件进行注册。这个处理程序指定了对话框作为每个工具提示控件的父窗口，并且为每个工具提示控件包含LPSTR_TEXTCALLBACK值。这样，在工具提示控件需要获取控件所需的文本时，对话框将接收到一个包含有TTN_NEEDTEXT通告消息的WM_NOTIFY消息。对话框处理程序调用OnWMNotify函数来处理TTN_NEEDTEXT通告消息。OnWMNotify函数能够根据工具提示控件标识符提供适当的字符串。

这个例子演示了如何使用TTM_RELAYEVENT向工具提示控件传递鼠标消息。为了能够访问鼠标消息，DoCreateDialogTooltip函数必须安装具有WH_GETMESSAGE数据类型的消息转发处理程序。消息转发处理程序GetMsgProc能够监视那些试图发送给其中一个控件窗口的鼠标消息流，并且将这些消息转发给工具提示控件。

```

//DoCreateDialogTooltip -creates a tooltip control for
//    a dialog box,enumerates the child control windows,
//    and installs a hook procedure to monitor the message
//    stream for mouse messages posted to the control
//    windows.
//Returns TRUE if successful,or FALSE otherwise.
//
//Global variables
//g_hinst -handle to the application instance

```

```

//g_hwndTT -handle to the tooltip control
//g_hwndDlg -handle to the dialog box
//g_hhk -handle to the hook procedure

BOOL DoCreateDialogTooltip(void)
{
    //Ensure that the common control DLL is loaded,
    //and create a tooltip control.
    InitCommonControls();
    g_hwndTT =CreateWindowEx(0,TOOLTIPS_CLASS,(LPSTR)NULL,
        TTS_ALWAYSSTIP,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
        CW_USEDEFAULT,g_hwndDlg,(HMENU)NULL,g_hinst,NULL);

    if (g_hwndTT ==NULL)
        return FALSE;

    //Enumerate the child windows to register them with
    //the tooltip control.
    if (!EnumChildWindows(g_hwndDlg,(WNDENUMPROC)EnumChildProc,0))
        return FALSE;

    //Install a hook procedure to monitor the message
    //stream for mouse messages intended for the controls
    //in the dialog box.
    g_hhk =SetWindowsHookEx(WH_GETMESSAGE,GetMagProc,
        (HINSTANCE)NULL,GetCurrentThreadId());

    if (g_hhk ==(HHOOK)NULL)
        return FALSE;

    return TRUE;
}

//EnumChildProc -registers control windows with a tooltip
// control by using the TTM_ADDTOOL message to pass the
// address of a TOOLINFO structure.
//Returns TRUE if successful,or FALSE otherwise.
//
//hWndCtrl -handle of a control window
//lParam -application-defined value (not used)
BOOL EnumChildProc(HWND hWndCtrl,LPARAM lParam)
{
    TOOLINFO ti;
    char szClass [64];

    //Skip static controls.
    GetClassName(hWndCtrl,szClass,sizeof(szClass));

```

```

    if (lstrcmpi(szClass, "STATIC"){
        ti.cbSize = sizeof(TOOLINFO);
        ti.uFlags = TTF_IDISHWND;
        ti.hwnd = g_hwndDlg;
        ti.uId = (UINT)hwndCtrl;
        ti.hinst = 0;
        ti.lpszText = LPSTR_TEXTCALLBACK;
        SendMessage(g_hwndTT, TTM_ADDTOOL, 0,
            (LPARAM) (LPTOOLINFO)&ti);
    }
    return TRUE;
}

//GetMsgProc -monitors the message stream for mouse
// messages intended for a control window in the
// dialog box.
//Returns a message-dependent value.
//
//nCode -hook code
//wParam -message flag (not used)
//lParam -address of an MSG structure
LRESULT CALLBACK GetMsgProc(int nCode, WPARAM wParam,
LPARAM lParam)
{
    MSG *lpmsg;

    lpmsg = (MSG *)lParam;
    if (nCode < 0 || !(IsChild(g_hwndDlg, lpmsg->hwnd)))
        return (CallNextHookEx(g_hhk, nCode, wParam, lParam));

    switch (lpmsg->message){
        case WM_MOUSEMOVE:
        case WM_LBUTTONDOWN:
        case WM_LBUTTONUP:
        case WM_RBUTTONDOWN:
        case WM_RBUTTONUP:
            if (g_hwndTT != NULL){
                MSG msg;

                msg.lParam = lpmsg->lParam;
                msg.wParam = lpmsg->wParam;
                msg.message = lpmsg->message;
                msg.hwnd = lpmsg->hwnd;
                SendMessage(g_hwndTT, TTM_RELAYEVENT, 0,
                    (LPARAM) (LPMSG)&msg);
            }
            break;
    }
}

```

```

        default:
            break;
    }
    return (CallNextHookEx(g_hhk,nCode,wParam,lParam));
}

//OnWMNotify -provides the tooltip control with the
// appropriate text to display for a control window.
// This function is called by the dialog box procedure
// in response to a WM_NOTIFY message.
//
//lParam -second message parameter of the WM_NOTIFY
// message.
VOID OnWMNotify(LPARAM lParam)
{
    LPTOOLTIPTEXT lpttt;
    int idCtrl;

    if (((LPNMHDR)lParam)->code)==TTN_NEEDTEXT){
        idCtrl =GetDlgCtrlID((HWND)((LPNMHDR)lParam)->idFrom);
        lpttt =(LPTOOLTIPTEXT)lParam;

        switch (idCtrl){
            case ID_HORZSCROLL:
                lpttt->lpszText ="A horizontal scroll bar.";
                return;

            case ID_CHECK:
                lpttt->lpszText ="A check box.";
                return;

            case ID_EDIT:
                lpttt->lpszText ="An edit control.";
                return;
        }
    }
    return;
}

```

25.3 在Internet Explorer中更新工具提示控件

Microsoft Internet Explorer中的工具提示控件支持两个新特性：跟踪工具提示与多行工具提示。

25.3.1 跟踪工具提示

工具提示控件能够支持跟踪工具提示功能，这种功能能够使得工具提示窗口在屏幕上动态

地改变自己的位置。由于工具提示窗口迅速地更新了自己的位置,从而使得它们看起来就像是平滑地移动一样,把这种平滑的移动称为“跟踪”。在希望工具提示文本能够随着鼠标的位置动态移动时,这个功能就非常有用了。

如果需要创建具有跟踪功能的工具提示,应该使用TTM_ADDTOOL消息,并且在向对应的TOOLINFO数据结构uFlags数据成员中设置TTF_TRACK标志位。

应用程序必须自行使用TTM_TRACKACTIVATE消息来激活或关闭具有跟踪功能的工具提示。在工具提示处于活跃状态时,应用程序必须使用TTM_TRACKPOSITION消息来提供工具提示窗口将要显示的位置信息。具有跟踪功能的工具提示控件并不支持TTF_SUBCLASS样式,因此所有的鼠标事件都必须通过TTM_RELAYEVENT消息从父窗口转发给子窗口。

TTM_TRACKPOSITION消息将会导致工具提示控件利用以下两种放置样式之一显示工具提示控件窗口:

- 缺省情况下,工具提示信息将显示在控件所选择的位置所在工具的旁边,而这个被选择的位置是相对于消息所提供的坐标而言的。在这种情况下,工具提示窗口将显示为向相应工具旁边移动。
- 如果在TOOLINFO数据结构uFlags数据成员中设置了TTF_ABSOLUTE标志位,那么工具提示将出现在由消息所指定的像素位置上。在这种情况下,工具提示控件并不会试图改变所提供的坐标上工具提示窗口的位置。

关于更多信息及其实现细节,请参见本节中的“创建跟踪工具提示”与“支持跟踪工具提示”。

1. 创建跟踪工具提示

下面的例子演示了如何创建工具提示控件以及如何向工具提示控件中添加一个工具。此例子将把主窗口的整个客户区声明为工具;当然,也可以声明客户区中的某个特定部分或者指定一个不同的窗口作为工具。

例子将使用TTM_ADDTOOL消息向工具提示控件中添加工具。由于跟踪工具提示并不支持TTF_SUBCLASS标志位,因此控件的所有者必须自行使用TTM_RELAYEVENT转发某些消息(例如WM_MOUSEMOVE消息)。

此外,这个例子中所使用的TOOLINFO数据结构uFlags数据成员中包含有TTF_ABSOLUTE标志位,这个标志位将会使得工具提示控件在发送TTM_TRACKPOSITION消息时,在应用程序所提供的坐标上显示工具提示文本。如果没有设置TTF_ABSOLUTE标志位,那么工具提示控件将根据所提供的坐标来选择将要显示工具提示文本的位置,从而导致工具提示文本出现在相应工具的旁边,但并不一定出现在由应用程序所提供的那个坐标上。

```
HWND WINAPI CreateTT(HWND hwndOwner)
{
    INITCOMMONCONTROLSEX icex;
    HWND      hwndTT;
    TOOLINFO  ti;

    //Load the tooltips class from the DLL.
    icex.dwSize = sizeof(icex);
```

```

icex.dwICC = ICC_BAR_CLASSES;

if(!InitCommonControlsEx(&icex))
    return NULL;

//Create the tooltip control.
hwndTT = CreateWindow(TOOLTIPS_CLASS, TEXT(""),
    WS_POPUP,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    NULL, (HMENU) NULL, g_hinst,
    NULL);

//Prepare the TOOLINFO structure to be used for a
// tracking tooltip.
ti.cbSize = sizeof(TOOLINFO);
ti.uFlags = TTF_IDISHWND | TTF_TRACK | TTF_ABSOLUTE;
ti.hwnd = hwndOwner;
ti.ulid = (UINT)g_hwndMain;
ti.hinst = g_hinst;
ti.lpszText = LPSTR_TEXTCALLBACK;
ti.rect.left = ti.rect.top = ti.rect.bottom = ti.rect.right = 0;

//Add the tool to the control, displaying an error
// if needed.
if(!SendMessage(hwndTT, TTM_ADDTOOL, 0, (LPARAM)&ti)){
    MessageBox(hwndOwner, "Couldn't create the tooltip
control.", "Error", MB_OK);
    return NULL;
}

//Activate (display) the tracking tooltip. Then, set
//a global flag value to indicate that the tooltip is
//active, so other functions can check to see if it's
//visible.
SendMessage(hwndTT, TTM_TRACKACTIVATE, (WPARAM) TRUE, (LPARAM)&ti);
g_bIsVisible = TRUE;
return(hwndTT);
}

```

2. 支持跟踪工具提示

下面的例子是一个简单的窗口处理函数，它能够支持跟踪工具提示功能。这个例子将使用 `GetCursorPos` 函数来获取鼠标的当前位置，然后为x坐标与y坐标增加15个像素点，从而使得工具提示显示在鼠标所在位置的右下方一点。

需要说明的是，这个例子将根据全局变量 `g_bIsVisible` 来判断应用程序是否应该发送 `TTM_TRACKPOSITION` 消息。为实现例子的演示功能，`g_bIsVisible` 全局变量应该是一个布尔

值，这个值由另一个函数在发送TTM_TRACKACTIVATE消息时设置为TRUE，从而激活工具提示。这样，如果工具提示处于非活跃状态，那么就可以避免用来计算与发送这个消息的额外时间开销。

```
LRESULT CALLBACK WndProc (HWND hwnd,UINT msg,WPARAM
wParam,LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    MSG msg;
    POINT pt;

    switch(msg){
        case WM_MOUSEMOVE:
            if(g_bIsVisible){
                msg.hwnd =g_hwndMain;
                msg.message =msg;
                msg.wParam =wParam;
                msg.lParam =lParam;

                GetCursorPos(&pt);
                msg.pt.x =pt.x;
                msg.pt.y =pt.y;

#define X_OFFSET 15
#define Y_OFFSET X_OFFSET
                SendMessage(g_hwndTT,
                    TTM_TRACKPOSITION,0,
                    (LPARAM)MAKELPARAM(pt.x+X_OFFSET,
                    pt.y+Y_OFFSET));
            }
            break;

        /*
        *
        *Other standard window messages can be handled here.
        *
        */
    }
    return 0;
}
```

25.3.2 多行工具提示

多行工具提示支持能够显示多行文本。如果需要显示的消息很长，并且不便于在一行之内进行阅读，那么多行提示工具功能就非常有用。图25-2显示了一个多行工具提示的例子，当鼠标移动到桌面的Internet Explorer图标上时，就显示了一个多行工具提示。

创建多行工具提示

应用程序在响应TTN_GETDISPINFO通告消息时,就能够创建多行工具提示。为了使工具提示控件使用多行工具提示的方法来显示文本,应该发送TTM_SETMAXTIPWIDTH消息,并且指定显示矩形的宽度。这时,当文本超过了所设置的宽度时,它将自动地转移到下一行进行显示,而不会加宽显示矩形的宽度。同时,显示矩形的高度将会根据显示多行代码的需要而发生改变。工具提示控件会自动地进行换行,也可以使用回车/换行组合“\r\n”,在某特定的位置强制进行换行。

需要说明的是,由NMTTDISPINFO数据结构中szText数据成员所指定的文本缓冲器只能容纳80字符。如果需要使用更长的字符串,那么就应该将NMTTDISPINFO数据结构的lpszText数据成员指向包含有所需文本的缓冲器。

下列代码段是一个简单的TTN_GETDISPINFO通告消息部分,它将显示矩形设置为300个像素,并且将NMTTDISPINFO数据结构的lpszText数据成员设置为指向带有所需文本的缓冲器,从而创建多行工具提示。

```
char szLongMessage [] =
    "This is a long message for the tooltip, which will "
    "automatically be wrapped when it exceeds the maximum "
    "tip width. Alternatively, you can use a \r \n carriage "
    "return/line -feed combination \r \n to force line breaks at "
    "specific \r \n locations.";

switch (lpmhdr->code){
    case TTN_GETDISPINFO:
        lpttd = (LPNMTTDISPINFO)lpmhdr;
        SendMessage(lpmhdr->hwndFrom, TTM_SETMAXTIPWIDTH, 0, 300);
        lpttd->lpszText = szLongMessage;
        return 0;

    ...
    //Other notification handlers go here, as needed.
}
```

25.3.3 球形工具提示

球形工具提示与标准的工具提示相似,但它们是卡通样式的“球形”进行显示的,并且

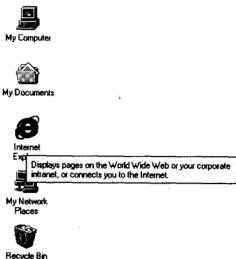


图25-2 多行工具提示

有一个指向工具的箭头，如图25-3所示。

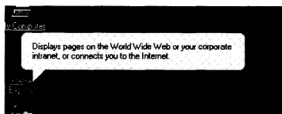


图25-3 “球形”工具提示

球形工具提示可以是单行的也可以是多行的，并且对它们的创建与处理与标准工具提示基本相同。球形工具提示的茎干缺省位置以及矩形形式如图25-3所示。如果工具过于靠近屏幕的顶部，那么工具提示将出现在此工具的下方，并且位于工具所在矩形的右方。如果工具过于靠近屏幕的右方，那么将适用类似的原则，工具提示将出现在工具矩形的左方。

可以设置工具提示TOOLINFO数据结构uFlags数据成员中的TTF_CENTERTIP标志位，从而改变缺省的显示位置。在这种情况下，工具提示的茎干通常指向工具矩形下边的中间部分，而所显示的文本矩形则位于工具的下方。同时，工具提示的茎干会紧贴在文本矩形上边界的中间部分。如果工具过于靠近屏幕的底部，那么文本矩形将显示在工具上方的中间，而工具提示的茎干则紧贴在工具提示下边界的中间部分。

如果需要指定工具提示茎干的指向，那么应该设置工具提示TOOLINFO数据结构uFlags数据成员中的TTF_TRACK标志位。然后，发送TTM_TRACKPOSITION消息来指定茎干应该指向的坐标，在这个消息的lParam参数中将指定x坐标与y坐标。如果设置了TTF_CENTERTIP标志位，那么茎干还将指向由TTM_TRACKPOSITION消息所指定的位置。

下列代码段演示如何实现居中放置的球形工具提示。

```
hwndToolTips = CreateWindow(TOOLTIPS_CLASS,
                             NULL,
                             WS_POPUP | TTS_NOPREFIX | TTS_BALLOON,
                             0, 0,
                             0, 0,
                             NULL, NULL,
                             g_hinst,
                             NULL);

if (hwndTooltip)
{
    //Do the standard tooltip coding.
    TOOLINFO ti;

    ti.cbSize = sizeof(ti);
    ti.uFlags = TTF_TRANSPARENT | TTF_CENTERTIP;
    ti.hwnd = hwnd;
    ti.uId = 0;
    ti.hinst = NULL;
```

```

ti.lpszText =LPSTR_TEXTCALLBACK;
GetClientRect(hwnd,&ti.rect);
SendMessage(hwndToolTips,TTM_ADDTOOL,0,(LPARAM)&ti );
}

```

状态条图标的球形工具提示

一个为状态条图标显示带有解释性消息的方法是实现一个球形工具提示，并且使得这个工具提示的箭头指向这个图标。当进行了鼠标单击时，工具提示将消失。当然，也可以为这个工具提示指定过期时间值。这种工具提示的外观如图25-4所示。

可以在NOTIFYICONDATA数据结构中设置NIF_INFO标志位，并且使用szInfo数据成员与uTimeout数据成员指定工具提示文本以及过期时间值。下列代码段演示了如何向状态条图标添加一个球形工具提示。

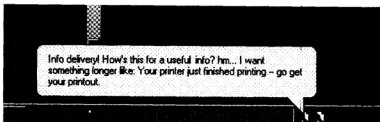


图25-4 状态条上的球形工具提示

```

NOTIFYICONDATA IconData =(0);

IconData.cbSize =sizeof(IconData);
IconData.hWnd =hwndNI;
IconData.uFlags =NIF_INFO;

lstrcpyn(IconData.szInfo,TEXT("Your message text goes here."),
ARRAYSIZE(IconData.szInfo));
IconData.uTimeout =15000;//in milliseconds

Shell_NotifyIcon(NIM_MODIFY,&IconData);

```

25.4 工具提示样式

除了标准的窗口样式之外，工具提示控件还支持许多控件样式。工具提示控件中具有WS_POPUP窗口样式与WS_EX_TOOLWINDOW窗口样式，而无论在创建此控件时是否指定了这两种样式。

工具提示控件可以使用下列控件样式：

- **TTS_ALWAYSSTIP** 这个样式用来指定当鼠标位于某个控件之上时，无论工具提示控件的所有者窗口是否活跃，工具提示控件都将出现。如果没有指定这个样式，工具提示将只在工具的所有者窗口活跃时才显示出来。

- **TTS_BALLOON** 在5.80版本中有效，这个样式用来指定工具提示控件具有卡通“球形”的外观，并且这个卡通球形的四个角为圆形的，而且球形的茎干指向项目。
- **TTS_NOANIMATE** 在5.80版本中有效，这个样式将关闭在Microsoft Windows 98系统与Microsoft Windows 2000系统中出现的滑动工具提示动画。在Windows早期版本的系统中，这个样式将被忽略。
- **TTS_NOFADE** 在5.80版本中有效，这个样式将关闭Windows 2000系统中的衰减工具提示动画。这个样式在Windows NT系统的早期版本以及Windows 95、Windows 98系统中被忽略。
- **TTS_NOPREFIX** 这个样式将能够阻止用户去除字符串中的“与”符号(&)。如果没有设置这个样式，那么系统将自动地去除字符串中所出现的&符号。这个样式能够使得应用程序使用相同的字符串作为菜单项目以及工具提示控件的文本。

25.5 工具提示控件参考

25.5.1 工具提示控件消息

TTM_ACTIVATE

激活或关闭工具提示控件。

```
TTM_ACTIVATE
    wParam = (WPARAM)(BOOL) fActivate;
    lParam = 0;
```

参数

fActivate: 激活标志。如果这个参数为TRUE，则表示工具提示控件被激活；否则，如果这个参数为FALSE，则表示工具提示控件被关闭。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_ADDTOOL

在工具提示控件中注册一个工具。

```
TTM_ADDTOOL
    wParam = 0;
    lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti: TOOLINFO数据结构的地址, 在这个数据结构中包含有工具提示控件需要用来为工具显示的文本信息。在发送此消息之前, 必须填充这个数据结构的cbSize数据成员。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_ADJUSTRECT

根据工具提示控件的窗口矩形, 计算工具提示控件的文本显示矩形, 或者计算需要显示指定文本显示矩形所需的工具提示窗口矩形。

```
TTM_ADJUSTRECT
wParam = (WPARAM)(BOOL) fLarger;
lParam = (LPARAM)(LPRECT) prc;
```

参数

fLarger: 这个参数用来说明应该执行的操作。如果这个参数为TRUE, 那么prc参数将被用来指定文本显示矩形, 并且接收相应的窗口矩形。如果这个参数为FALSE, 那么prc参数将被用来指定一个窗口矩形, 并且接收相应的文本显示矩形。

prc: RECT数据结构, 这个数据结构用来存放工具提示窗口矩形或文本显示矩形。

返回值

如果成功地改变了矩形的大小, 则返回非零值; 否则, 如果在执行过程中出现了错误, 则返回0。

说明

在需要使用工具提示控件来显示通常被截取的字符串的所有文本时, 这个消息特别有用。此消息通常与列表视图控件与树视图控件一同使用。在响应TTM_SHOW通告消息时, 就需要发送这个消息, 从而能够正确地对工具提示控件进行定位。

工具提示的窗口矩形一般比包围在工具提示字符串周围的文本显示矩形大一点, 而且工具提示窗口矩形通常在文本显示矩形的上端与左端。为了确定文本显示矩形的位置, 必须计算相应的窗口矩形, 并且使用这个窗口矩形来确定工具提示的位置。上述计算操作可以利用TTM_ADJUSTRECT来完成。

如果将fLarger参数设置为TRUE, 那么TTM_ADJUSTRECT将获取所需工具提示文本显示矩形的大小与位置, 并且传递工具提示窗口的大小与位置值用来在指定的位置显示文本。如果将fLarger参数设置为FALSE, 那么就可以指定工具提示的窗口矩形, 并且TTM_ADJUSTRECT将返回工具提示文本矩形的大小与位置。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_DELTOOL

从工具提示控件中删除一个工具。

TTM_DELTOOL

wParam = 0;

lParam = (LPARAM)(LPTOOLINFO) lpti;

参数

lpti: TOOLINFO数据结构的地址。其中的hwnd数据成员与uld数据成员用来标识将要被删除的工具, cbSize数据成员用来指定数据结构的大小, 其他的数据成员将被忽略。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_ENUMTOOLS

获取工具提示控件为当前工具所维护的信息, 这里所说的当前工具是指工具提示控件目前正在为其显示文本的那个工具。

TTM_ENUMTOOLS

wParam = (WPARAM)(UINT) iTool;

lParam = (LPARAM)(LPTOOLINFO) lpti;

参数

iTool: 将要从中获取信息的工具基于0的索引。

lpti: TOOLINFO数据结构的地址, 这个数据结构可用来获取关于此工具的信息。在发送这个消息之前, cbSize数据成员必须指定了此数据结构的大小。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_GETBUBBLESIZE

返回工具提示控件的宽度与高度。

```
TTM_GETBUBBLESIZE
    wParam = 0;
    lParam = (LPARAM)(LPTOOLINFO) pTtm;
```

参数

pTtm: 指向工具提示的TOOLINFO数据结构的指针。

返回值

如果操作成功, 那么在返回值的低位字部分存放工具提示的宽度值, 在返回值的高位字部分存放工具提示的高度字; 否则, 返回FALSE。

说明

如果设置了工具提示TOOLINFO数据结构和uFlags数据成员中的TTF_TRACK与TTF_ABSOLUTE标志位, 那么这个消息就可以用来帮助确定工具提示的位置。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_GETCURRENTTOOL

获取工具提示控件中当前工具的信息。

```
TTM_GETCURRENTTOOL
    wParam = 0;
    lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti: TOOLINFO数据结构的地址, 这个数据结构用来接收关于当前工具的信息。如果这个参数值为NULL, 那么返回值将说明当前工具的存在性, 但不返回工具的信息。如果这个值不是NULL, 那么必须在发送这个消息之前填充TOOLINFO数据结构的cbSize数据成员。

返回值

如果操作成功，则返回非零值；否则，返回0。如果lpti参数是NULL，那么如果当前工具存在，则返回非零值；否则，返回0。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

TTM_GETDELAYTIME

获取工具提示控件的初始时间间隔、弹出时间间隔以及重显时间间隔的当前设置值。

```
TTM_GETDELAYTIME
    wParam = (DWORD) dwDuration;
    lParam = 0;
```

参数

dwDuration：这个参数用来说明将要获取哪个时间间隔值，它可以下列值之一：

- | | |
|--------------|---|
| TTDT_AUTOPOP | 如果鼠标指针一直静态地位于工具的约束矩形之内，那么就获取工具提示窗口维持可见状态条的时间长度。 |
| TTDT_INITIAL | 在工具提示窗口出现之前，鼠标指针必须在工具的约束矩形中维持静止不动状态时间长度。 |
| TTDT_RESHOW | 当鼠标指针从一个工具移到另一个工具上时，后一个工具的工具提示窗口出现所需要的时间间隔。 |

返回值

返回以毫秒为单位的时间间隔INT值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

TTM_SETDELAYTIME。

TTM_GETMARGIN

获取为工具提示窗口设置的上下左右页边空白。所谓页边空白是指工具提示窗口边界与包

含在工具提示窗口中的文本之间的距离，单位为像素。

```
TTM_GETMARGIN
    wParam = 0;
    lParam = (LPARAM)(LPRECT) lprc;
```

参数

lprc: RECT数据结构的地址，这个数据结构用来接收页边空白信息。

RECT数据结构的数据成员并不定义约束矩形。在这个消息中，此数据结构的数据成员将作如下解释：

top (顶部页边空白)	表示顶部边界与工具提示文本顶部之间的距离，单位为像素。
left (左边页边空白)	表示左边界与工具提示文本左端之间的距离，单位为像素。
bottom (底部页边空白)	表示底部边界与工具提示文本底部之间的距离，单位为像素。
right (右边页边空白)	表示右边界与工具提示文本右端之间的距离，单位为像素。

返回值

这个消息的返回值没有被使用。

说明

在创建工具提示控件时，所有的这四个页边空白都缺省地设置为0。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_SETMARGIN。

TTM_GETMAXTIPWIDTH

获取工具提示窗口的最大宽度。

```
TTM_GETMAXTIPWIDTH
    wParam = 0;
    lParam = 0;
```

返回值

返回用来表示工具提示最大宽度的INT值，单位为像素。如果当前没有设置最大宽度，那么这个消息将返回-1。

说明

最大工具提示宽度值并不能说明工具提示窗口的实际宽度。相反，如果工具提示字符串超过了工具提示的最大宽度，那么工具提示控件将把文本分隔成多行进行显示，并且使用空格来

作为进行分行的标志。如果文本不能被分隔成多行，那么这些文本将在单独的一行上进行显示。这样，这行文本的宽度就可能会超过最大工具提示宽度。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

TTM_SETMAXTIPWIDTH。

TTM_GETTEXT

获取工具提示控件为某个工具所维护的信息。

```
TTM_GETTEXT
wParam = 0;
lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti：TOOLINFO数据结构的地址。

在发送这个消息之前，必须填充此数据结构的cbSize数据成员。设置hwnd数据成员与uld数据成员来标识将要从其中获取信息的工具。设置lpszText数据成员指向将要获取文本的缓冲器。目前还没有方法能够指定缓冲器的大小或判断所需的缓冲器大小。

返回值

没有返回值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TTM_GETTIPBKCOLOR

获取工具提示窗口的背景色。

```
TTM_GETTIPBKCOLOR
wParam = 0;
lParam = 0;
```

返回值

返回用来表示背景色的COLORREF值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_SETTIPBKCOLOR。

TTM_GETTOOLCOUNT

获取由工具提示控件所维护的工具的数目。

```
TTM_GETTOOLCOUNT
```

```
wParam = 0;
```

```
lParam = 0;
```

返回值

返回工具的数目。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_GETTOOLINFO

获取工具提示控件为某个工具所维护的信息。

```
TTM_GETTOOLINFO
```

```
wParam = 0;
```

```
lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti: TOOLINFO数据结构的地址。在发送这个消息时, hwnd数据成员与Id数据成员用来标识工具, cbSize数据成员必须指定数据结构的大小。如果工具提示控件包含有这个工具, 那么此数据结构将获取本工具的信息。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_HITTEST

测试某个点, 判断这个点是否在指定工具的约束矩形之内。如果在约束矩形之内, 则获取关于这个工具的信息。

```
TTM_HITTEST  
wParam = 0;  
lParam = (LPARAM)(LPHITTESTINFO) lphti;
```

参数

lphti: TTHITTESTINFO数据结构的地址。在发送这个消息时, hwnd数据成员必须指定某个工具的句柄, pt数据成员必须指定将要进行测试的点的坐标。如果返回值为TRUE, 那么ti数据成员(它是一个TOOLINFO数据结构)将接收到位于这个点上的工具的信息。在发送这个消息之前必须填充ti数据结构的cbSize数据成员。

返回值

如果指定的点在工具之上, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_NEWTOOLRECT

为某个工具设置新的约束矩形。

```
TTM_NEWTOOLRECT  
wParam = 0;  
lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti: TOOLINFO数据结构的地址。其中, hwnd数据成员与uld数据成员用来标识一个工具, rect数据成员用来指定新的约束矩形。在发送这个消息之前, 必须填充此数据结构的cbSize数据成员。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_POP

从视图中删除一个被显示的工具提示窗口。

```
TTM_POP
    wParam = 0;
    lParam = 0;
```

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_RELAYEVENT

向工具提示控件传递一个鼠标消息, 从而处理这个鼠标消息。

```
TTM_RELAYEVENT
    wParam = 0;
    lParam = (LPARAM)(LPMSG) lpmsg;
```

参数

lpmsg: MSG数据结构的地址, 在这个数据结构中包含有将要被转发的消息。

返回值

没有返回值。

说明

工具提示控件只能处理由TTM_RELAYEVENT消息传递给它的下列消息:

- WM_LBUTTONDOWN
- WM_LBUTTONUP
- WM_MBUTTONDOWN
- WM_MBUTTONUP
- WM_MOUSEMOVE
- WM_RBUTTONDOWN
- WM_RBUTTONUP

其他所有的消息将被忽略。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_SETDELAYTIME

设置工具提示控件的初始时间间隔、弹出时间间隔以及重显时间间隔。

```
TTM_SETDELAYTIME
wParam = (WPARAM)(DWORD) dwDuration;
lParam = (LPARAM)(INT) MAKELONG(iTime, 0);
```

参数

dwDuration: 这个参数用来说明将要被设置的时间间隔值, 它可以是下列值之一:

- | | |
|----------------|---|
| TTDT_AUTOMATIC | 将这三个延时时间设置为其缺省值。其中, 自动弹出时间间隔是初始时间间隔的10倍, 重显时间间隔是初始时间间隔的五分之一。如果设置了这个标志位, 那么就必须使用iTime参数的一个正值来指定初始时间间隔, 单位为毫秒。如果将iTime参数设置为一个负值, 那么分别将初始时间间隔、弹出时间间隔以及重显时间间隔设置为缺省的500ms、5000ms以及100ms。 |
| TTDT_AUTOPOP | 如果鼠标指针一直静态地位于工具的约束矩形之内, 那么就获取工具提示窗口维持可见状态条的时间长度。 |
| TTDT_INITIAL | 在工具提示窗口出现之前, 鼠标指针必须在工具的约束矩形中维持静止不动状态的时间长度。 |
| TTDT_RESHOW | 当鼠标指针从一个工具移到另一个工具上时, 后一个工具的工具提示窗口出现所需要的时间间隔。 |

iTime: 延迟时间值, 单位为毫秒。

返回值

这个消息的返回值没有被使用。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_GETDELAYTIME。

TTM_SETMARGIN

为工具提示控件设置上下左右页边空白。所谓页边空白是指工具提示窗口边界与包含在工

具提示窗口中的文本之间的距离, 单位为像素。

```
TTM_SETMARGIN
    wParam = 0;
    lParam = (LPARAM)(LPRECT) lprc;
```

参数

lprc: RECT数据结构的地址, 这个数据结构中包含有将要被设置的页边空白信息。

RECT数据结构的数据成员并不定义约束矩形。在这个消息中, 此数据结构的数据成员将作如下解释:

top (顶部页边空白)	表示顶部边界与工具提示文本顶部之间的距离, 单位为像素。
left (左边页边空白)	表示左边界与工具提示文本左端之间的距离, 单位为像素。
bottom (底部页边空白)	表示底部边界与工具提示文本底部之间的距离, 单位为像素。
right (右边页边空白)	表示右边界与工具提示文本右端之间的距离, 单位为像素。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_GETMARGIN。

TTM_SETMAXTIPWIDTH

设置工具提示窗口的最大宽度。

```
TTM_SETMAXTIPWIDTH
    wParam = 0;
    lParam = (LPARAM)(INT) iWidth;
```

参数

iWidth: 将要被设置的最大工具提示子窗口宽度。

返回值

返回用来表示先前最大工具提示宽度的INT值。

说明

最大工具提示宽度值并不能说明工具提示窗口的实际宽度。相反, 如果工具提示字符串超过了工具提示的最大宽度, 那么工具提示控件将把文本分隔成多行进行显示, 并且使用空格来作为进行分行的标志。如果文本不能被分隔成多行, 那么这些文本将在单独的一行上进行显示。

这样，这行文本的宽度就可能会超过最大工具提示宽度。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_GETMAXTIPWIDTH。

TTM_SETTIPBKCOLOR

设置工具提示窗口的背景色。

TTM_SETTIPBKCOLOR

```
wParam = (WPARAM)(COLORREF) clr;
```

```
lParam = 0;
```

参数

clr: 新的背景色。

返回值

这个值返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_GETTIPBKCOLOR。

TTM_SETTIPTEXTCOLOR

设置工具提示窗口的文本颜色。

TTM_SETTIPTEXTCOLOR

```
wParam = (WPARAM)(COLORREF) clr;
```

```
lParam = 0;
```

参数

clr: 新的文本颜色。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

TTM_GETTIPTEXTCOLOR。

TTM_SETTITLE

向工具提示添加一个标准图标与标题字符串。

```
TTM_SETTITLE  
    wParam = icon;,  
    lParam = (LPCTSTR) pszTitle;
```

参数

icon: 将wParam设置为下列值之一, 用来指定将要被显示的图标:

- 0 表示没有图标需要显示。
- 1 表示显示信息图标。
- 2 表示显示警告图标。
- 3 表示显示出错图标。

pszTitle: 指向标题字符串的指针, 必须为pszTitle设置一个值。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

需要使用动态链接库Comctl32.dll的5.80版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 5.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_SETTOOLINFO

设置工具提示控件为某个工具所维护的信息。

```
TTM_SETTOOLINFO
wParam = 0;
lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti: TOOLINFO数据结构的地址, 这个数据结构用来指定将要被设置的信息。在发送这个消息之前, 必须填充此数据结构的cbSize数据成员。

返回值

没有返回值。

说明

工具的某些内部属性是在工具被创建时所建立的, 并且在TTM_SETTOOLINFO消息被发送时不会被重新计算。如果为TOOLINFO数据结构设置值, 并且利用TTM_SETTOOLINFO消息将这个数据结构传递给工具提示控件, 那么这些属性将被丢失。相反, 应用程序应该向工具提示控件发送TTM_GETTOOLINFO消息, 从而请求控件的当前TOOLINFO数据结构。然后, 按照需要修改这个数据结构的相关数据成员, 并且利用TTM_SETTOOLINFO将这些数据成员传递给工具提示控件。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_TRACKACTIVATE

激活或关闭跟踪工具提示。

```
TTM_TRACKACTIVATE
wParam = (WPARAM)(BOOL) bActivate;
lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

bActivate: 这个参数用来指定跟踪工具提示功能被激活还是被关闭, 它可以是下列值之一:

FALSE 表示关闭跟踪工具提示功能。

TRUE 表示激活跟踪工具提示功能。

lpti: TOOLINFO数据结构的地址, 这个数据结构用来说明这个消息所应用的工具。其中, **hwnd**数据成员与**uld**数据成员用来标识工具, **cbSize**数据成员用来指定数据结构的大小。其他所有的数据成员将被忽略。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

25.3.1节、TTM_TRACKPOSITION。

TTM_TRACKPOSITION

设置跟踪工具提示的位置。

TTM_TRACKPOSITION

wParam = 0;

lParam = (LPARAM)(DWORD)MAKELONG(xPos, yPos);

参数

xPos和yPos: 分别表示跟踪工具提示将要被显示的点的x坐标与y坐标, 在屏幕坐标下。

返回值

这个消息的返回值没有被使用。

说明

工具提示控件将根据这个消息所提供的坐标来选择在何处显示工具提示窗口, 这样将会导致工具提示窗口出现在与工具提示相对应的工具旁边。如果需要使工具提示窗口在指定的坐标上显示, 那么应该在添加这个工具时, 在TOOLINFO数据结构的uFlags数据成员中设置TTF_ABSOLUTE标志位。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

25.3.1节、TTM_TRACKACTIVATE。

TTM_UPDATE

强制当前工具进行重绘。

```
TTM_UPDATE
    wParam = 0;
    lParam = 0;
```

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_UPDATETIPTTEXT

为某个工具设置工具提示文本。

```
TTM_UPDATETIPTTEXT
    wParam = 0;
    lParam = (LPARAM)(LPTOOLINFO) lpti;
```

参数

lpti: TOOLINFO数据结构的地址。其中, hinst数据成员与lpszText数据成员必须指定句柄实例以及文本的地址, hwnd数据成员与uld数据成员用来指定将要被更新的工具。在发送这个消息之前, 此数据结构的cbSize数据成员必须被填充。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTM_WINDOWFROMPOINT

这个消息将使得子类处理程序为不是位于鼠标之下的另一个窗口显示工具提示文本。

```
TTM_WINDOWFROMPOINT
    wParam = 0;
    lParam = (POINT FAR *) lppt;
```

参数

lppt: POINT数据结构的地址, 这个数据结构用来定义将要被检查的点。

返回值

所有的返回值是包含有这个点的窗口句柄，如果没有任何窗口在指定的点上，那么将返回NULL。

说明

这个消息可以被作为工具提示子类的应用程序所处理，而不是被某个应用程序发送。某个工具提示在为窗口显示文本之前，将向自身发送这个消息。如果改变由lppt所指定点的坐标，那么子类处理程序将使得工具提示为不是在鼠标之下的那个窗口显示文本。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

25.5.2 工具提示控件通告消息**NM_CUSTOMDRAW (工具提示)**

这个通告消息由工具提示控件发送，用来告知工具提示控件的父窗口所执行的绘制操作。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CUSTOMDRAW
    lpNMCustomDraw = (LPNMTCUSTOMDRAW) lParam;
```

参数

lpNMCustomDraw：指向NMTCUSTOMDRAW数据结构的指针，在这个数据结构中包含有关于绘制操作的信息。

返回值

应用程序所能够返回的值取决于当前绘制状态。在与这个通告消息相关联的NMCUSTOMDRAW数据结构dwDrawStage数据成员中包含有用来指定绘制阶段的值。必须返回下列值之一：

当dwDrawStage等于CDDS_PREPAINT时：

CDRF_DODEFAULT：控件将绘制自身。在这个绘制阶段，工具提示控件不会发送任何其他的NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYITEMDRAW：控件将告知父窗口任何与项目相关的绘制操作。它将在绘制项目之前与之后发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYITEMERASE：当某个项目将要被擦除时，控件将告知父窗口。控件将在擦除项目之前与之后发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYPOSTERASE：控件将在擦除某个项目之后告知父窗口。

CDRF_NOTIFYPOSTPAINT：控件将在绘制某个项目之后告知父窗口。

CDRF_NOTIFYSUBITEMDRAW：在4.71版本中有效，控件将在正在绘制某个列表视图子

项目时向父窗口发送这个通告消息。

当dwDrawStage等于CDDS_ITEMPREPAINT时:

CDRF_NEWFONT: 表示应用程序为项目指定了一个新的字体, 并且控件将使用新字体。

CDRF_SKIPDEFAULT: 表示应用程序将自行绘制项目, 这时控件就不会绘制此项目。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

8.3节。

TTN_GETDISPINFO

这个通告消息由工具提示控件发送, 用来获取显示工具提示窗口所需的信息。此通告消息正在逐步取代TTN_NEEDTEXT通告消息, 它是以WM_NOTIFY消息的形式进行发送的。

```
TTN_GETDISPINFO
    lpnmtdi = (LPNMTTDISPINFO) lParam;

#define TTN_NEEDTEXT TTN_GETDISPINFO
```

参数

lpnmtdi: NMTTDISPINFO数据结构的地址, 这个数据结构用来定义需要文本的工具以及获取所需的信息。

返回值

这个通告消息的返回值没有被使用。

说明

为了能够为工具提示控件返回所需的信息, 必须首先填充数据结构的适当域。如果消息处理程序将NMTTDISPINFO数据结构的uFlags域设置为TTF_DI_SETITEM, 那么工具提示控件将存储信息并且不会再次请求此信息。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTN_POP

这个通告消息用来告知所有者窗口, 工具提示将要被隐藏。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
TTN_POP
    idTT = (int) wParam;
    pnmh = (LPMNHDR) lParam;
```

参数

idTT: 工具提示控件的标识符。

pnmh: NMHDR数据结构的地址。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTN_SHOW

这个通告消息用来告知所有者窗口, 工具提示控件将要被显示。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
TTN_SHOW
    idTT = (int) wParam;
    pnmh = (LPMNHDR) lParam;
```

参数

idTT: 工具提示控件的标识符。

pnmh: NMHDR数据结构的地址。

返回值

在4.70版本中有效, 如果将要在缺省位置显示工具提示, 返回0。如果需要自定义工具提示的位置, 那么必须利用SetWindowPos函数来重新确定工具提示窗口的位置, 并且返回TRUE。

注意 对于4.70之前的版本, 没有返回值。

说明

工具提示的窗口矩形一般比文本显示矩形大, 并且前者在后者的左上方。如果需要精确地确定工具提示的文本显示矩形, 那么就应该使用TTM_ADJUSTRECT消息将文本显示矩形转换为相应的工具提示窗口矩形。反之亦然。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

25.5.3 工具提示控件数据结构**NMTTCUSTOMDRAW**

在这个数据结构中包含有由工具提示控件所发送的NM_CUSTOMDRAW通告消息所特有的信息。

```
typedef struct tagNMTTCUSTOMDRAW {
    NMCUSTOMDRAW nmcd;
    UINT          uDrawFlags;
} NMTTCUSTOMDRAW, FAR * LPNMTTCUSTOMDRAW;
```

成员

nmcd: 这个数据成员是一个NMCUSTOMDRAW数据结构, 其中包含有常用的定制绘图信息。

uDrawFlags: 这个数据成员是一个UINT值, 用来说明工具提示文本在被显示时应该使用何种格式。应用程序可能会改变这个域, 从而修改文本的显示方式。这个数据成员的值是在内部传递给DrawText函数, DrawText的所有uFormat参数值都是合法的。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

NMTTDISPINFO

在这个数据结构中包含有处理TTN_GETDISPINFO通告消息时所使用的信息。这个数据结构正在逐渐取代TOOLTIPTEXT数据结构。

```
typedef struct tagNMTTDISPINFO {
    NMHDR          hdr;
    LPTSTR          lpszText;
    char            szText [80 ];
    HINSTANCE       hinst;
```



```

        UINT            uFlags;
    #if (_WIN32_IE >= 0x0300)
        LPARAM          lParam;
    #endif
    }NMTTDISPINFO, FAR *LPNMTTDISPINFO;

    #define TOOLTIPTTEXT NMTTDISPINFO

```

成员

hdr: 这个数据成员是NMHDR数据结构，在其中包含有关于本通告消息的更多信息。

lpzText: 这个数据成员是以null结尾的字符串的地址，这个字符串包含有将要被显示作为工具提示文本的信息。如果hinst数据成员指定了一个实例句柄，那么这个数据成员就必须是字符串资源的标识符。

szText: 用来接收工具提示文本的缓冲器。应用程序可以不指定字符串地址或字符串资源，而是将文本拷贝到这个缓冲器中。对于多于80个字符的工具提示文本，请参见本节中的说明部分。

hinst: 包含有将要被用作工具提示文本的字符串资源的实例句柄。如果lpzText是工具提示文本字符串的地址，那么这个数据成员必须为NULL。

uFlags: 这个数据成员用来说明如何解释包含在NMHDR数据结构中的idFrom数据成员。

TTF_IDISHWND 如果设置了这个标志位，那么idFrom数据成员就是工具的句柄。否则，就是工具的标识符。

TTF_RTLREADING 如果设置了这个标志位，那么表示窗口可以被镜像，从而显示诸如希伯来语或阿拉伯语之类按照从右到左（RTL）进行读写的语言。一般情况下，工具提示文本是按照与其父窗口相同的方向进行读写的。如果需要将工具提示文本显示位于其父窗口相反的方向，那么就应该在处理这个通告消息时为uFlags数据成员设置TTF_RTLREADING标志位。

TTF_DI_SETITEM 在4.70版本中有效，如果在处理这个通告消息时为uFlags数据成员设置了这个标志位，那么工具提示控件将仍然维持所提供的信息，从而不必再次请求此信息。

lParam: 在4.70版本中有效，这个数据成员表示与工具相关联的、由应用程序所定义的数据。

说明

如果用来在工具提示中显示的文本超过了80个字符，那么就必须使得lpzText数组指向自行定义的一个私有缓冲器。除非控件使用了TTS_NOPREFIX样式，否则系统将自动地从传递给工具提示控件的所有字符串中提取出“与”符号(&)加速字符。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TOOLINFO

在这个数据成员中包含有关于工具提示控件中某个工具的信息。

```
typedef struct tagTOOLINFO{
    UINT        cbSize;
    UINT        uFlags;
    HWND        hwnd;
    WPARAM      uId;
    RECT        rect;
    HINSTANCE    hinst;
    LPTSTR       lpszText;
    #if (_WIN32_IE >=0x0300)
        LPARAM    lParam;
    #endif
} TOOLINFO, NEAR *PTOOLINFO, FAR *LPTOOLINFO;
```

成员

cbSize: 这个数据成员用来表示此数据结构的大小, 单位为字节。此外, 这个数据成员必须被设置值。

uFlags: 这个数据成员可用来控制工具提示的显示, 它可以是下列值的组合:

TTF_ABSOLUTE 在4.70版本中有效, 用来定位与由TTM_TRACKPOSITION所提供的坐标相同的点上的工具提示窗口。这个标志位必须与TTF_TRACK标志位一同使用。

TTF_CENTERTIP 将由uld数据成员所指定的工具下面的工具提示窗口居中。

TTF_IDISHWND 这个标志位用来说明uld数据成员就是工具的窗口句柄。如果这个标志位没有被设置, 那么uld数据成员就表示工具的标识符。

TTF_RTREADING 这个标志位用来说明工具提示文本将按照与父窗口相反的文本显示方向进行显示。

TTF_SUBCLASS 为了实现对诸如WM_MOUSEMOVE消息之类的消息进行监听, 这个标志位用来指定工具提示控件应该设置工具窗口的子类。如果没有设置这个标志位, 那么必须使用TTM_RELAYEVENT消息向工具提示控件转发消息。如果需要了解工具提示控件所处理的消息列表, 请参见TTM_RELAYEVENT。

TTF_TRACK 在4.70版本中有效, 定位于工具相对应的工具提示窗口, 并且根据TTM_TRACKPOSITION消息中所提供的坐标移动窗口。必须使用TTM_TRACKACTIVATE消息激活这种类型的工具。

TTF_TRANSPARENT 在4.70版本中有效, 这个标志位将使得工具提示控件将鼠标事

件消息转发给父窗口。这些鼠标事件消息仅仅限于在工具提示窗口的约束矩形中所发生的鼠标事件。

hwnd: 包含有本工具的窗口的句柄。如果lpszText中包含有LPSTR_TEXTCALLBACK值, 那么这个数据成员标识了将要接收TTN_GETDISPINFO通告消息的那个窗口。

uld: 应用程序为工具所定义的标识符。如果uFlags数据成员中包含有TTF_IDISHWND标志位, 那么uld必须指定工具的窗口句柄。

rect: 工具的约束矩形坐标。这些坐标是相对于由hwnd所定义的窗口的客户区左上角的坐标。如果uFlags数据成员中包含有TTF_IDISHWND标志位, 那么这个数据成员将被忽略。

hinst: 包含有工具的字符串资源实例的句柄。如果lpszText指定了字符串资源的标识符, 那么这个数据成员将被使用。

lpszText: 指向包含有工具文本的缓冲器的指针, 或者表示包含有文本的字符串资源的标识符。有时, 这个数据成员用来返回值。如果需要检查返回值, 那么lpszText数据成员必须指向一个合法的、具有足够大小的缓冲器。否则, 这个数据成员应该被设置为NULL。如果lpszText数据成员被设置为LPSTR_TEXTCALLBACK, 那么控件将向所有者窗口发送TTN_NEEDTEXT通告消息, 从而接收所需的信息。

lParam: 在4.70版本中有效, 用来表示一个与工具相关联的、由应用程序所定义的32位值。

说明

通常, 窗口是按照从左到右(LTR)的方式来显示文本。当然, 窗口可以被镜像, 用来显示诸如希伯来语或阿拉伯语之类按照从右到左(RTL)的方式进行读写的语言。一般情况下, 工具提示文本是按照与其父窗口文本相同方向进行显示的。如果设置了TTF_RTREADING标志位, 那么工具提示文本将按照与其父窗口相反的方向进行显示。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TTHITTESTINFO

这个数据结构中包含有工具提示控件用来判断某个点是否在指定工具的约束矩形中的信息。如果这个点在约束矩形中, 那么此数据结构将接收关于本工具的信息。

```
typedef struct TT_HITTESTINFO {
    HWND    hwnd;
    POINT   pt;
    TOOLINFO ti;
} TTHITTESTINFO, FAR *LPHITTESTINFO;
```

成员

hwnd: 工具的句柄或由工具所指定的窗口的句柄。

pt: 将要被测试的点的客户坐标。

ti: 这个数据成员是一个TOOLINFO数据结构。如果由pt数据成员所指定的点在由hwnd数据成员所指定的工具之上, 那么这个数据结构将接收关于本工具的信息。在发送这个消息之前, 此数据结构的cbSize数据成员必须被填充。

说明

这个数据结构必须与TTM_HITTEST消息一同使用。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

发送TBM_SETTICFREQ消息。例如，可以使用这个消息来在1到100范围之内只显示10个计数标记。

如果需要设置一个计数标记的位置，那么应该发送TBM_SETTIC消息。轨迹条维持有一个DWORD数组，这个数组中存储了每个计数标记的位置。当然，在这个数组中并不包含轨迹条自动创建的第一个计数标记与最后一个计数标记。在发送TBM_GETTIC消息来获取相应计数标记的位置时，就应该指定这个计数标记相对应的数组元素的索引。也可以发送TBM_GETPTICS消息来获取数组的指针。数组中的元素数目等于由TBM_GETNUMTICS所返回的计数标记数目减2。这是因为TBM_GETNUMTICS所返回的计数标记数目中包含有在数组中没有包含的第一个计数标记与最后一个计数标记。如果需要获取计数标记在轨迹条窗口客户区坐标中的物理位置，那么就应该发送TBM_GETTICPOS消息。TBM_CLEARICS消息可用来删除除了第一个计数标记与最后一个计数标记之外的其他计数标记。

轨迹条的线长决定了在响应来自箭头键（例如向右箭头键与向下箭头键）的键盘输入时或者响应鼠标输入时，滑标应该移动的距离。可以分别使用TBM_GETLINESIZE消息与TBM_SETLINESIZE消息来获取与设置线长。在用户按下箭头键时，轨迹条还将发送TB_LINEUP通告消息与TB_LINEDOWN通告消息来告知父窗口。

轨迹条的页面大小决定了在响应来自键盘输入（例如向上翻页键与向下翻页键）或者响应鼠标输入（例如在轨迹条轨道上单击鼠标）时，滑标将要移动的距离。可以分别使用TBM_GETPAGESIZE消息与TBM_SETPAGESIZE消息来获取与设置页面大小。在轨迹条接收用来滚动页面的键盘输入或鼠标输入时，将向轨迹条父窗口发送TB_PAGEUP通告消息与TB_PAGEDOWN通告消息。关于更多信息，请参见26.1.2节“轨迹条通告消息”。

应用程序可以向轨迹条发送消息来获取轨迹条的大小。其中，TBM_GETTHUMBRECT消息可用来获取滑标的约束矩形。TBM_GETTHUMBLENGTH消息可用来获取滑标的长度。TBM_GETCHANNELRECT消息可用来获取轨迹条轨道的约束矩形，所谓轨迹条轨道，就是滑标能够在其上移动的那个区域，在某个范围值被选中时，它将显示为高亮状态。如果轨迹条具有TBS_FIXEDLENGTH样式，那么可以发送TBM_SETTHUMBLENGTH消息来改变滑标的长度。

如果使用TBS_ENABLESELRANGE样式创建轨迹条，就可以指定“选择范围”，从而限制用户在全部轨迹条范围的某个部分进行选择。这时，逻辑单元并不会改变，但只有逻辑单元的一个子集可以被使用。轨迹条对可以被使用的范围进行高亮显示，并且在这个范围之内的起始位置与结束位置时间显示三角形的计数标记。典型情况下，应用程序将根据用户输入来处理轨迹条的通告消息并设置轨迹条的选择范围。

可以向轨迹条发送消息，来设置或选择轨迹的选择范围。其中，TBM_SETSEL消息可用来设置选择范围的起始位置与结束位置。如果只需要设置选择范围的起始位置或设置选择范围的结束位置，那么可以使用TBM_SETSELSTART消息或TBM_SETSELEND消息。如果需要获取选择范围的起始位置与结束位置，那么就应该发送TBM_GETSELSTART消息或TBM_GETSELEND消息。如果需要清除所设置的选择范围并且将轨迹条恢复到其原始范围，那么就应该发送TBM_CLEARSEL消息。

26.1.2 轨迹条通告消息

轨迹条将向其父窗口发送WM_HSCROLL消息或WM_VSCROLL消息，用来告知父窗口用户所执行的动作。其中，具有TBS_HORZ样式的轨迹条将发送WM_HSCROLL消息，具有TBS_VERT样式的轨迹条则发送WM_VSCROLL消息。WM_HSCROLL消息与WM_VSCROLL消息的wParam参数低位字部分包含有通告消息代码。对于TB_THUMBPOSITION通告消息与TB_THUMBTRACK通告消息，那么在wParam参数的高位字部分将指定滑标位置。对于其他所有的通告消息，这个参数的高位字部分将为0。发送TBM_GETPOS消息可用来判断滑标位置。lParam参数是轨迹条的句柄。

只有在用户使用键盘与轨迹条进行交互时，系统才发送TB_BOTTOM通告消息、TB_LINEDOWN通告消息、TB_LINEUP通告消息与TB_TOP通告消息。只有在用户使用鼠标与轨迹条进行交互时，系统才发送TB_THUMBPOSITION通告消息与TB_THUMBTRACK通告消息。在两种情况下，系统都将会发送TB_ENDTRACK通告消息、TB_PAGEDOWN通告消息与TB_PAGEUP通告消息。下表列出了轨迹条通告消息以及导致通告消息被发送的事件（虚拟键代码或鼠标事件）：

通告消息	发送通告消息的原因
TB_BOTTOM	VK_END
TB_ENDTRACK	WM_KEYUP（用户释放了一个将要发送虚拟键代码的键）
TB_LINEDOWN	VK_RIGHT或VK_DOWN
TB_LINEUP	VK_LEFT或VK_UP
TB_PAGEDOWN	VK_NEXT（用户单击了轨迹条轨道的下方或单击了滑标的右方）
TB_PAGEUP	VK_PRIOR（用户单击了轨迹条轨道的上方或单击了滑标的左方）
TB_THUMBPOSITION	在TB_THUMBTRACK通告消息之后发送了WM_LBUTTONDOWN通告消息
TB_THUMBTRACK	滑标进行了移动（例如用户拖动了滑标）
TB_POP	VK_HOME

26.1.3 缺省轨迹条消息处理

本节描述轨迹条所处理的窗口消息。

消 息	执行的处理动作
WM_CAPTURECHANGED	如果在处理WM_LBUTTONDOWN消息的过程中设置了计时器，那么就删除这个计时器，并且在必要时发送TB_THUMBPOSITION通告消息。然后发送TB_ENDTRACK通告消息
WM_CREATE	执行更多的初始化操作，例如设置线长、页面大小、计数标记频率设置为缺省值等等
WM_DESTROY	释放资源
WM_ENABLE	重绘轨迹条窗口
WM_ERASEBKGD	利用轨迹条的当前背景色，擦除窗口背景
WM_GETDLGCODE	返回DLGC_WANTARROWS值

(续)

消 息	执行的处理动作
WM_KEYDOWN	处理方向键,并且按照需要发送TB_TOP、TB_BOTTOM、TB_PAGEUP、TB_PAGEDOWN、TB_LINEUP与TB_LINEDOWN通告消息
WM_KEYUP	如果键是一个方向键,则发送TB_ENDTRACK通告消息
WM_KILLFOCUS	重绘轨迹条窗口
WM_LBUTTONDOWN	为轨迹条设置焦点并且设置鼠标捕获。在必要时,还可以设置一个计时器,用来决定当用户在窗口中一直按下鼠标按钮时,滑标向鼠标进行移动的速度
WM_LBUTTONUP	释放鼠标捕获,并且删除计时器(如果在处理WM_LBUTTONDOWN消息时设置了计时器)。必要时,它将发送TB_THUMBPOSITION通告消息。这个消息总是会发送TB_ENDTRACK通告消息
WM_MOUSEMOVE	在跟踪鼠标时(请参见WM_TIMER),移动滑标并发送TB_THUMBTRACK通告消息
WM_PAINT	绘制轨迹条。如果wParam参数不是NULL,那么控件将假设这个值是一个HDC并且使用此设备上下文绘制轨迹条
WM_SETFOCUS	重绘轨迹条窗口
WM_SIZE	设置轨迹条的大小,如果没有足够的大小,滑标将被删除
WM_TIMER	获取鼠标位置并且更新滑标的位置(只有在用户拖动滑标时,才接收这个消息)
WM_WININICHANGE	初始设置滑标大小

26.2 使用轨迹条控件

本节将给出用来演示如何创建轨迹条以及处理轨迹条通告消息的例子。

26.2.1 创建轨迹条

下面的例子演示了如何创建带有TBS_AUTOTICKS样式与TBS_ENABLESELRANGE样式的轨迹条。在创建这个轨迹条时,此轨迹条的范围以及选择范围都被设置,而且它的页面大小也被设置。

```
//CreateTrackbar -creates and initializes a
// trackbar.
//
//Global variable
// g_hinst -instance handle
HWND WINAPI CreateTrackbar(
    HWND hwndDlg, //handle of dialog box (parent window)
    UINT iMin,    //minimum value in trackbar range
    UINT iMax,    //maximum value in trackbar range
    UINT iSelMin, //minimum value in trackbar selection
    UINT iSelMax) //maximum value in trackbar selection
{
```



```

InitCommonControls();//loads common control's DLL

hwndTrack =CreateWindowEx(
    0,                                //no extended
    styles
    TRACKBAR_CLASS,                    //class name
    'Trackbar Control',                //title (caption)
    WS_CHILD |WS_VISIBLE |
    TBS_AUTOTICKS |TBS_ENABLESELRANGE, //style
    10,10,                              //position
    200,30,                             //size
    hwndDlg,                           //parent window
    ID_TRACKBAR,                        //control identifier
    g_hinst,                           //instance
    NULL                                //no WM_CREATE
parameter
);

SendMessage(hwndTrack,TBM_SETRANGE,
    (LPARAM)TRUE,                      //redraw flag
    (LPARAM)MAKELONG(iMin,iMax));      //min.& max.
positions
SendMessage(hwndTrack,TBM_SETPAGESIZE,
    0,(LPARAM)4);                     //new page size

SendMessage(hwndTrack,TBM_SETSEL,
    (LPARAM)FALSE,                     //redraw flag
    (LPARAM)MAKELONG(iSelMin,iSelMax);
SendMessage(hwndTrack,TBM_SETPOS,
    (LPARAM)TRUE,                      //redraw flag
    (LPARAM)iSelMin);

SetFocus(hwndTrack);

return hwndTrack;
}

```

26.2.2 处理轨迹条通告消息

下面的例子是一个函数，当包含有轨迹条的对话框接收到WM_HSCROLL消息时，将调用这个函数。轨迹条具有TBS_ENABLESELRANGE样式，在这个例子中，滑标的位置将与选择范围的位置进行比较，并且在必要时，滑标可以在选择范围的起始位置与结束位置之间进行移动。

包含有TBS_VERT样式轨迹条的对话框在接收到WM_VSCROLL消息时可以调用这个函数。

```

//TBNNotifications -handles trackbar
// notifications received in the wParam
// parameter of WM_HSCROLL.This function

```

```

// simply ensures that the slider remains
// within the selection range.

VOID WINAPI TBNotifications(
WPARAM wParam, //wParam of WM_HSCROLL message
HWND hwndTrack, //handle of trackbar window
UINT iSelMin, //minimum value of trackbar selection
UINT iSelMax) //maximum value of trackbar selection
{
DWORD dwPos; //current position of slider

switch (LOWORD(wParam)){
case TB_ENDTRACK:
dwPos =SendMessage(hwndTrack,TBM_GETPOS,0,
0);

if (dwPos >iSelMax)
SendMessage(hwndTrack,TBM_SETPOS,
(WPARAM)TRUE, //redraw flag
(LPARAM)iSelMax);
else if (dwPos <iSelMin)
SendMessage(hwndTrack,TBM_SETPOS,
(WPARAM)TRUE, //redraw flag
(LPARAM)iSelMin);

break;

default:
break;
}
}

```

1. CDRF_NEWFONT

表示应用程序为项目定义了新字体，并且控件将使用这个新字体。

2. CDRF_SKIPDEFAULT

表示应用程序将自行绘制项目，因此控件不用绘制项目。

26.3 在Internet Explorer中更新轨迹条控件

Microsoft Internet Explorer中的轨迹条控件支持下列新特性。

1. Buddy Windows

轨迹条控件现在可以支持两个伴随窗口。轨迹条控件将自动地确定轨迹条伴随窗口的位置，并且使它们出现在轨迹条控件的尾部居中位置。如果需要将一个现有窗口分配给轨迹条，就应该使用TBM_SETBUDDY消息。如果需要获取某个给定伴随窗口的句柄，就应该使用TBM_GETBUDDY消息。

2. Tooltips

轨迹条控件现在可以支持工具提示功能。在创建TBS_TOOLTIPS样式的轨迹条时，将会创建缺省的工具提示控件。但是，也可以发送TBM_SETTOOLTIPS消息，为轨迹条分配一个新的工具提示控件。如果需要获取被分配的工具提示控件的句柄，就应该使用TBM_GETTOOLTIPS消息。

26.4 轨迹条控件样式

本节将讨论轨迹条控件所使用的样式。

1. TBS_AUTOTICKS

在这种样式下，轨迹条控件将在自己范围值之内的每个增加值上有一个计数标记。

2. TBS_BOTH

在这种样式下，轨迹条控件将在控件的两边都显示计数标记。对于TBS_HROZ样式的轨迹条控件，也就是在轨迹条的上方与下方都显示计数标记；对于TBS_VERT样式的轨迹条控件，也就是在轨迹条的左边与右边都显示计数标记。

3. TBS_BOTTOM

在这个样式下，轨迹条控件将在控件的下边显示计数标记。这个样式仅仅对TBS_HROZ样式控件有效。

4. TBS_ENABLESELRANGE

在这种样式下，轨迹条控件将只显示选择范围。其中，选择范围起始位置与结束位置上的计数标记被显示为三角形（而不是竖直线），并且选择范围被高亮显示。

5. TBS_FIXEDLENGTH

在这种样式下，轨迹条控件将允许TBM_SETTHUMBLENGTH消息改变滑标的大小。

6. TBS_HORZ

在这种样式下，轨迹条控件将使用竖直方向。这是缺省的轨迹条方向。

7. TBS_LEFT

在这种样式下，轨迹条控件将在控件的左边显示计数标记。这个样式仅仅对TBS_VERT样式轨迹条控件有效。

8. TBS_NOTHUMB

在这种样式下，轨迹条控件将不显示滑标。

9. TBS_NOTICKS

在这种样式下，轨迹条控件将不显示任何计数标记。

10. TBS_REVERSED

在5.80版本中有效，这个样式用来实现“倒转”轨迹条。在这种轨迹条中，当所设置的值小于轨迹条的下限值时，那么这个值将用来表示轨迹条的一个“较大”值；当所设置的值大于轨迹条的上限值时，那么这个值将用来表示轨迹条的一个“较小”值。这种样式对于控件没有任何影响，它只是用来标明轨迹条是普通轨迹条还是“倒转”轨迹条。

11. TBS_RIGHT

在这种样式下，轨迹条控件将在控件的右边显示计数标记。这个样式仅仅对于TBS_VERT样式轨迹条控件有效。

12. TBS_TOOLTIPS

在4.70版本中有效，在这种样式下，轨迹条控件将支持工具提示功能。当利用这个样式创建一个轨迹条控件时，系统将会自动地创建一个缺省的工具提示控件，用来显示滑标的当前位置。可以使用TBM_SETTIPSIDE消息来改变工具提示被显示的位置。

13. TBS_TOP

在这种样式下，轨迹条控件将在控件的上边显示计数标记。这个样式仅仅对于TBS_HORZ样式轨迹条控件有效。

14. TBS_VERT

在这种样式下，轨迹条控件将使用竖直方向。这是缺省的轨迹条方向。

26.5 定制绘图值

轨迹条控件使用下列值来标识控件的各个部分。其中的一个值将由NMCUSTOMDRAW数据结构中的dwItemSpec数据成员来指定。

- TBCD_CHANNEL 这个值用来标识工具条控件的滑标在其上进行移动的轨道。这个部分是用户可以移动的控件部分。
- TBCD_THUMB 这个值用来标识轨迹条控件的标记。
- TBCD_TICS 这个值用来标识沿着轨迹条控件边所显示的计数标记。

26.6 轨迹条控件参考

26.6.1 轨迹条控件消息

TBM_CLEARSEL

清除轨迹条中的当前选择范围。

```
TBM_CLEARSEL
wParam = (WPARAM)(BOOL) fRedraw;
lParam = 0;
```

参数

fRedraw: 重绘标志位。如果这个参数为TRUE，那么表示轨迹条将在选择范围被清除之后进行重绘。

返回值

没有返回值。

说明

只有在创建轨迹条时指定了TBS_ENABLESELRANGE样式的情况下，轨迹条才会拥有选择范围。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_SETSEL、TBM_SETSELEND、TBM_SETSELSTART。

TBM_CLEARITICS

删除轨迹条中的当前计数标记。这个消息并不会删除轨迹条自动创建的第一个计数标记与最后一个计数标记。

```
TBM_CLEARITICS
wParam = (WPARAM)(BOOL) fRedraw;
lParam = 0;
```

参数

fRedraw: 重绘标志位。如果这个参数为TRUE, 那么表示轨迹条将在计数标记被删除之后进行重绘; 如果这个参数为FALSE, 那么表示在删除计数标记之后不对轨迹条进行重绘。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETBUDDY

获取给定位置上轨迹条控件伴随窗口的句柄。在这里, 所指定的位置是相对于控件的方向(水平还是竖直)来定义的。

```
TBM_GETBUDDY
wParam = (WPARAM)(BOOL) fLocation;
lParam = 0;
```

参数

fLocation: 这个参数用来说明将要获取哪个伴随窗口的句柄, 基于相对位置而言。它可以是下列值之一:

- TRUE** 表示获取轨迹条左边伴随窗口的句柄。如果轨迹条控件使用了TBS_VERT样式, 那么这个消息将获取轨迹条上面伴随窗口的句柄。
- FALSE** 表示获取轨迹条右边伴随窗口的句柄。如果轨迹条控件使用了TBS_VERT样式, 那么这个消息将获取轨迹条下面伴随窗口的句柄。

返回值

返回由fLocation所指定的位置上的伴随窗口句柄, 如果在这个位置上不存在任何伴随窗口, 则返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版

本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 需要使用Windows CE 2.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETCHANNELRECT

获取轨迹条轨道约束矩形的大小与位置。所谓轨道是指轨迹条的滑标在其上移动的区域, 当某个范围被选中时, 将显示高亮形式。

```
TBM_GETCHANNELRECT
wParam = 0;
lParam = (LPARAM)(LPRECT) lprc;
```

参数

lprc: RECT数据结构的地址。消息将利用轨道约束矩形的信息填充这个数据结构, 所使用的坐标系是约束条窗口的客户区坐标。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETLINESIZE

获取轨迹条控件在响应来自箭头键的键盘输入(例如向右箭头键或向下箭头键)时, 轨迹条滑标所移动的逻辑位置数目。所谓逻辑位置是指在轨迹条最大滑标位置与最小滑标位置区域之间的整型增量值位置。

```
TBM_GETLINESIZE
wParam = 0;
lParam = 0;
```

返回值

返回用来表示轨迹条线长的32位值。

说明

线长的缺省设置值为1。

当用户按下箭头键时, 轨迹条将向其父窗口发送TB_LINEUP通告消息与TB_LINEDOWN通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_SETLINESIZE。

TBM_GETNUMTICS

获取轨迹条中计数标记的数目。

```
TBM_GETNUMTICS
    wParam = 0;
    lParam = 0;
```

返回值

返回计数标记的数目。

说明

TBM_GETNUMTICS消息将对所有的计数标记进行统计, 包括有轨迹条所创建的第一个计数标记与最后一个计数标记。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETPAGESIZE

获取轨迹条控件在响应键盘输入(例如向上翻页键或向下翻页键)或者在响应鼠标输入(例如在轨迹条轨道中进行了鼠标单击)时, 轨迹条滑标所移动的逻辑位置数目。所谓逻辑位置是指在轨迹条最大滑标位置与最小滑标位置之间的整型增量位置。

```
TBM_GETPAGESIZE
    wParam = 0;
    lParam = 0;
```

返回值

返回用来表示轨迹条页面大小的32位值。

说明

轨迹条在接收到进行页面滚动的键盘输入或鼠标输入时, 也将会向其父窗口发送TB_PAGEUP通告消息与TB_PAGEDOWN通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_SETPAGESIZE。

TBM_GETPOS

获取轨迹条中滑标当前的逻辑位置。这个逻辑位置应该是轨迹条范围之内最小滑标位置到最大滑标位置之间的一个整型值。

TBM_GETPOS

wParam = 0;

lParam = 0;

返回值

返回用来表示轨迹条滑标当前逻辑位置的32位值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_SETPOS。

TBM_GETPTICS

获取包含有轨迹条计数标记位置信息数组的地址。

TBM_GETPTICS

wParam = 0;

lParam = 0;

返回值

返回一个DWORD数组地址。这个数组中的元素分别用来指定轨迹条计数标记的逻辑位置,但其中并不包括由轨迹条所创建的第一个计数标记与最后一个计数标记。逻辑位置可以是轨迹条最小滑标位置到最大滑标位置之间的任何一个整型值。

说明

这个数组中的元素数目比TBM_GETNUMTICS消息所返回的计数标记中的值少2。需要说明的是,数组中的这些值可能会包含重复的位置值,并且可能不是按照大小顺序进行排列的。直到改变轨迹条计数标记之前,这个数组中的指针都是有效的。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETRANGEMAX

获取轨迹条中滑标的最大位置。

```
TBM_GETRANGEMAX  
wParam = 0;  
lParam = 0;
```

返回值

返回用来表示轨迹条滑标可到达的最大位置的32位值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETRANGEMIN、TBM_SETRANGE、TBM_SETRANGEMAX。

TBM_GETRANGEMIN

获取轨迹条中滑标的最小位置。

```
TBM_GETRANGEMIN  
wParam = 0;  
lParam = 0;
```

返回值

返回用来表示轨迹条滑标可到达的最小位置的32位值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETRANGEMAX、TBM_SETRANGE、TBM_SETRANGEMAX。

TBM_GETSELEND

获取轨迹条当前选择范围的结束位置。

```
TBM_GETSELEND  
wParam = 0;  
lParam = 0;
```

返回值

返回用来表示轨迹条当前选择范围结束位置的32位值。

说明

只有在创建轨迹条控件时指定了TBS_ENABLESELRANGE样式，轨迹条才可能会拥有选择范围。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

TBM_GETSELSTART、TBM_SETSEL、TBM_SETSELEND、TBM_SETSELSTART。

TBM_GETSELSTART

获取轨迹条当前选择范围的开始位置。

TBM_GETSELSTART

wParam = 0;

lParam = 0;

返回值

返回用来表示轨迹条当前选择范围开始位置的32位值。

说明

只有在创建轨迹条控件时指定了TBS_ENABLESELRANGE样式，轨迹条才可能会拥有选择范围。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

TBM_GETSELEND、TBM_SETSEL、TBM_SETSELEND、TBM_SETSELSTART。

TBM_GETTHUMBLENGTH

获取轨迹条中滑标的长度。

TBM_GETTHUMBLENGTH

wParam = 0;

lParam = 0;

返回值

返回滑标的长度，单位为像素。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETTHUMBRECT、TBM_SETTHUMBLENGTH。

TBM_GETTHUMBRECT

获取轨迹条中滑标约束矩形的大小与位置。

```
TBM_GETTHUMBRECT
    wParam = 0;
    lParam = (LPARAM)(LPRECT) lprc;
```

参数

lprc: RECT数据结构的地址。这个消息将利用轨迹条滑标的约束矩形信息来填充此数据结构, 所使用的坐标系是轨迹条窗口的客户区坐标。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETTIC

获取轨迹条中计数标记的逻辑位置。逻辑位置可以是轨迹条最小滑标位置到最大滑标位置之间的任何整型值。

```
TBM_GETTIC
    wParam = (WPARAM)(WORD) iTic;
    lParam = 0;
```

参数

iTic: 计数标记基于0的索引。计数标记合法的索引必须在0到由TBM_GETNUMTICS消息所返回的计数标记总数减去2之间。

返回值

返回指定计数标记的逻辑位置, 如果iTic参数没有指定合法的计数标记索引, 则返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETTICPOS

获取轨迹条中某个计数标记的当前物理位置。

```
TBM_GETTICPOS
wParam = (WPARAM)(WORD) iTic;
lParam = 0;
```

参数

iTic: 计数标记基于0的索引。计数标记合法的索引必须在0到由TBM_GETNUMTICS消息所返回的计数标记总数减去2之间。

返回值

返回客户区坐标下的距离, 这个距离是从轨迹条客户区的左边或顶部到达指定计数标记的距离。对于水平轨迹条, 返回值是计数标记的x坐标; 对于竖直轨迹条, 返回值是计数标记的y坐标。如果iTic参数不是一个合法的索引, 那么返回-1。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

TBM_GETTOOLTIPS

获取分配给轨迹条的工具提示控件(若存在)的句柄。

```
TBM_GETTOOLTIPS
wParam = 0;
lParam = 0;
```

返回值

返回分配给轨迹条的工具提示控件的句柄, 如果没有使用工具提示控件, 则返回NULL。如果轨迹条控件没有使用TBS_TOOLTIPS样式, 则返回NULL。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000(或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98(或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TBM_GETUNICODEFORMAT

获取控件的UNICODE字符格式标志。

```
TBM_GETUNICODEFORMAT
wParam = 0;
lParam = 0;
```

返回值

返回控件UNICODE格式标志。如果所得的返回值为非零，那么表示控件在使用UNICODE字符；如果所得的返回值为0，那么表示控件在使用ANSI字符。

说明

关于这个消息的更多讨论，请参见CCM_GETUNICODEFORMAT。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

TBM_SETUNICODEFORMAT。

TBM_SETBUDDY

分配一个窗口作为轨迹条控件的伴随窗口。轨迹条的伴随窗口将根据控件的方向（水平还是竖直）自动地在某个位置上进行显示。

```
TBM_SETBUDDY
wParam = (WPARAM)(BOOL) fLocation;
lParam = (LPARAM)(HWND) hwndBuddy;
```

参数

fLocation：这个参数用来指定将显示伴随窗口的位置。它可以是下列值之一：

- TRUE** 这个值表示如果轨迹条控件使用了TBS_HORZ样式，那么伴随窗口将显示在轨迹条的左边。如果轨迹条控件使用了TBS_VERT样式，那么伴随窗口将显示在轨迹条控件的上面。
- FALSE** 这个值表示如果轨迹条控件使用了TBS_HORZ样式，那么伴随窗口将显示在轨迹条的右边。如果轨迹条控件使用了TBS_VERT样式，那么伴随窗口将显示在轨迹条控件的下面。

hwndBuddy：要被设置为轨迹条控件伴随窗口的窗口句柄。

返回值

返回先前在这个位置上分配给轨迹条控件的窗口句柄。

说明

注意：轨迹条控件能够支持两个伴随窗口。如果需要在轨迹条控件的两端显示文本或图像，那么这个特点就非常有用。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 3.0或更新版本的Windows 95）。

Windows CE：需要使用Windows CE 2.0或更新版本。

头文件：在commctrl.h中进行了声明。

TBM_SETLINESIZE

设置轨迹条控件在响应来自箭头键的键盘输入（例如向右箭头键或向下箭头键）时，轨迹条滑标所移动的逻辑位置数目。所谓逻辑位置是指在轨迹条最大滑标位置与最小滑标位置区域之间的整型增量值位置。

```
TBM_SETLINESIZE
    wParam = 0;
    lParam = (LONG) lLineSize;
```

参数

lLineSize：新的线长值。

返回值

返回用来表示先前线长的32位值。

说明

线长的缺省设置值为1。

当用户按下箭头键时，轨迹条将向其父窗口发送TB_LINEUP通告消息与TB_LINEDOWN通告消息。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

TBM_GETLINESIZE。

TBM_SETPAGESIZE

设置轨迹条控件在响应键盘输入（例如向上翻页键或向下翻页键）或者在响应鼠标输入（例如在轨迹条轨道中进行了鼠标单击）时，轨迹条滑标所移动的逻辑位置数目。所谓逻辑位置

是指在轨迹条最大滑标位置与最小滑标位置区域之间的整型增量值位置。

```
TBM_SETPAGESIZE
    wParam = 0;
    lParam = (LONG) lPageSize;
```

参数

lPageSize: 新的页面大小值。

返回值

返回用来表明先前页面大小的32位值。

说明

轨迹条在接收到进行页面滚动的键盘输入或鼠标输入时，也将会向其父窗口发送TB_PAGEUP通告消息与TB_PAGEDOWN通告消息。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETPAGESIZE。

TBM_SETPOS

设置轨迹条中滑标当前的逻辑位置。

```
TBM_SETPOS
    wParam = (WPARAM) (BOOL) fPosition;
    lParam = (LPARAM) (LONG) lPosition;
```

参数

fPosition: 重绘标志位。如果这个参数为TRUE, 表示消息将在由lPosition参数所指定的位置上重绘轨迹条控件与其滑标。如果这个参数为FALSE, 表示消息并不会在新位置重绘滑标。需要说明的是, 这个消息将设置滑标的位置(根据由TBM_GETPOS消息所返回的值进行设置), 而不考虑fPosition参数的影响。

lPosition: 滑标的新逻辑位置。合法的逻辑位置应该是在轨迹条最小滑标位置到最大滑标位置之间的整型值。如果这个值超出了控件的最大范围或最小范围, 那么这个位置将被相应地设置为最大值或最小值。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

TBM_SETRANGE

为轨迹条中的滑标设置最大位置与最小位置。

TBM_SETRANGE

```
wParam = (WPARAM) fRedraw;
lParam = (LPARAM) MAKELONG(lMinimum, lMaximum);
```

参数

fRedraw：重绘标志位。如果这个参数为TRUE，那么表示轨迹条将在被设置范围之后进行重绘。如果这个参数为FALSE，那么表示消息将设置范围但不进行轨迹条重绘。

lMinimum：滑标的最小位置。

lMaximum：滑标的最大位置。

返回值

没有返回值。

说明

如果当前滑标位置超出了新的范围，那么TBM_SETRANGE消息将把滑标位置设置为新的最大范围值或最小范围值。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

TBM_SETRANGEMAX、TBM_SETRANGEMIN。

TBM_SETRANGEMAX

为轨迹条中的滑标设置最大逻辑位置。

TBM_SETRANGEMAX

```
wParam = (WPARAM) fRedraw;
lParam = (LPARAM) lMaximum;
```

参数

fRedraw：重绘标志位。如果这个参数为TRUE，那么表示轨迹条将在被设置范围之后进行重绘。如果这个参数为FALSE，那么表示消息将设置范围但不进行轨迹条重绘。

lMaximum：滑标的最大位置。

返回值

没有返回值。

说明

如果当前滑标位置大于新的最大位置值，那么TBM_SETRANGEMAX消息将把滑标位置设

置为这个新的最大位置值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_SETRANGE、TBM_SETRANGEMIN。

TBM_SETRANGEMIN

为轨迹条中的滑标设置最小逻辑位置。

TBM_SETRANGEMIN

```
wParam = (WPARAM) fRedraw;
lParam = (LPARAM) lMinimum;
```

参数

fRedraw: 重绘标志位。如果这个参数为TRUE, 那么表示轨迹条将在被设置范围之后进行重绘。如果这个参数为FALSE, 那么表示消息将设置范围但不进行轨迹条重绘。

lMinimum: 滑标的最小位置。

返回值

没有返回值。

说明

如果当前滑标位置小于新的最小位置值, 那么TBM_SETRANGEMIN消息将把滑标位置置为这个新的最小位置值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_SETRANGE、TBM_SETRANGEMAX。

TBM_SETSEL

为轨迹条中的可用选择范围设置开始位置与结束位置。

TBM_SETSEL

```
wParam = (WPARAM) (BOOL) fRedraw;
lParam = (LPARAM) MAKELONG (lMinimum, lMaximum);
```

参数

fRedraw: 重绘标志位。如果这个参数为TRUE, 那么表示轨迹条将在被设置选择范围之后

进行重绘。如果这个参数为FALSE, 那么表示消息将设置选择范围但不进行轨迹条重绘。

lMinimum: 选择范围的开始逻辑位置。

lMaximum: 选择范围的结束逻辑位置。

返回值

没有返回值。

说明

如果轨迹条没有TBS_ENABLESELRange样式, 那么这个消息将被忽略。

TBM_SETSEL消息可以用来将鼠标指针限定在进度条的某个可用范围之内。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETSELEND、TBM_GETSELSTART、TBM_SETSELEND、TBM_SETSELSTART。

TBM_SETSELEND

设置轨迹条中当前选中范围的结束逻辑位置。如果轨迹条控件没有指定TBS_ENABLESELRange样式, 那么这个消息将被忽略。

TBM_SETSELEND

wParam = (WPARAM)(BOOL) fRedraw;

lParam = (LPARAM)(LONG) lEnd;

参数

fRedraw: 重绘标志位。如果这个参数为TRUE, 那么表示轨迹条将在被设置选择范围之后进行重绘。如果这个参数为FALSE, 那么表示消息将设置选择范围但不进行轨迹条重绘。

lEnd: 选择范围的结束逻辑位置。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETSELEND、TBM_GETSELSTART、TBM_SETSEL、TBM_SETSELSTART。

TBM_SETSELSTART

设置轨迹条中当前选中范围的开始逻辑位置。如果轨迹条控件没有指定TBS_

ENABLESELRange样式, 那么这个消息将被忽略。

```
TBM_SETSELSTART
wParam = (WPARAM)(BOOL) fRedraw;
lParam = (LPARAM)(LONG) lStart;
```

参数

fRedraw: 重绘标志位。如果这个参数为TRUE, 那么表示轨迹条将在被设置选择范围之后进行重绘。如果这个参数为FALSE, 那么表示消息将设置选择范围但不进行轨迹条重绘。

lStart: 选择范围的开始逻辑位置。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETSELEND、TBM_GETSELSTART、TBM_SETSEL、TBM_SETSELEND。

TBM_SETTHUMBLENGTH

设置轨迹条中滑标的长度。如果轨迹条控件没有使用TBS_FIXEDLENGTH样式, 那么这个消息将被忽略。

```
TBM_SETTHUMBLENGTH
wParam = (WPARAM)(UINT) iLength;
lParam = 0;
```

参数

iLength: 滑标的长度, 单位为像素。

返回值

没有返回值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETTHUMBLENGTH。

TBM_SETTIC

在某个给定的位置设置轨迹条的计数标记。

```
TBM_SETTIC
    wParam = 0;
    lParam = (LPARAM)(LONG) lPosition;
```

参数

lPosition: 计数标记的位置。这个参数可以是轨迹条最小滑标位置到最大滑标位置之间的任何整型值。

返回值

如果设置了计数标记，则返回TRUE；否则，返回FALSE。

说明

轨迹条能够自动地创建自己的第一个计数标记与最后一个计数标记。因此不能使用这个消息来设置一个计数标记与最后一个计数标记。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

TBM_GETPTICS、TBM_GETTIC。

TBM_SETTIPSIDE

设置轨迹条控件所使用的工具提示控件的位置。具有TBS_TOOLTIPS样式的轨迹条控件将显示工具提示控件。

```
TBM_SETTIPSIDE
    wParam = (WPARAM)(int) fLocation;
    lParam = 0;
```

参数

fLocation: 用来表示将要在哪里显示工具提示控件的位置，这个值可以是下列值之一：

TBTS_TOP	工具提示控件将位于轨迹条的上面。这个标志位对于水平轨迹条有效。
TBTS_LEFT	工具提示控件将位于轨迹条的左边。这个标志位对于竖直轨迹条有效。
TBTS_BOTTOM	工具提示控件将位于轨迹条的下面。这个标志位对于水平轨迹条有效。
TBTS_RIGHT	工具提示控件将位于轨迹条的右边。这个标志位对于竖直轨迹条有效。

返回值

返回用来表示工具提示控件当前位置的值，这个返回值应该是fLocation参数中的一个可能值。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

TBM_SETTOOLTIPS

向轨迹条控件分配一个工具提示控件。

```
TBM_SETTOOLTIPS
    wParam = (WPARAM)(HWND) hwndTT;
    lParam = 0;
```

参数

hwndTT: 现有工具提示控件的句柄。

返回值

这个消息的返回值没有被使用。

说明

当使用TBS_TOOLTIPS样式创建轨迹条控件时, 它将自动地创建一个缺省的工具提示控件, 并且这个工具提示控件出现在滑标的旁边, 用来显示滑标的当前位置。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

26.6.2 轨迹条控件通告消息**NM_CUSTOMDRAW (轨迹条)**

这个通告消息由轨迹条控件发送, 用来告知父窗口相关的绘制操作。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_CUSTOMDRAW
    lpNMCustomDraw = (LPNMCUSTOMDRAW) lParam;
```

参数

lpNMCustDraw: NMCUSTOMDRAW数据结构的地址, 在这个数据结构中包含有关于绘制操作信息。其中的dwItemSpec数据成员包含有一个定制绘图值, 用来说明控件的哪个部分正在被绘制。轨迹条控件将向dwItemSpec数据成员中插入以下值, 用来标识正在被绘制的控件部分:

- TBCD_CHANNEL** 这个值用来标识轨迹条控件滑标正在移动的那个轨道。
- TBCD_THUMB** 这个值用来标识轨迹条控件的滑标, 这个部分也就是用户能够移动的那个控件部分。
- TBCD_TICS** 用来标识出现在轨迹条控件边上的计数标记。

返回值

应用程序所能够返回的值取决于当前绘制状态。在与这个通告消息相关联的NMCUSTOMDRAW数据结构dwDrawStage数据成员中包含有用来指定绘制阶段的值。必须返回下列值之一:

当dwDrawStage等于CDDS_PREPARE时:

CDRF_DODEFAULT: 控件将绘制自身。在这个绘制阶段, 工具提示控件不会发送任何其他NMCUSTOMDRAW通告消息。

CDRF_NOTIFYITEMDRAW: 控件将告知父窗口任何与项目相关的绘制操作。它将在绘制项目之前与之后发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYITEMERASE: 当某个项目将要被擦除时, 控件将告知父窗口。控件将在擦除项目之前与之后发送NM_CUSTOMDRAW通告消息。

CDRF_NOTIFYPOSTERASE: 控件将在擦除某个项目之后告知父窗口。

CDRF_NOTIFYPOSTPAINT: 控件将在绘制某个项目之后告知父窗口。

CDRF_NOTIFYSUBITEMDRAW: 在4.71版本中有效, 控件将在正在绘制某个列表视图子项目时向父窗口发送这个通告消息。

当dwDrawStage等于CDDS_ITEMPREPARE时:

CDRF_NEWFONT: 表示应用程序为项目指定了一个新的字体, 并且控件将使用新字体。

CDRF_SKIPDEFAULT: 表示应用程序将自行绘制项目, 这时控件就不会绘制此项目。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

8.3节。

NM_RELEASEDCAPTURE (轨迹条)

这个通告消息用来告知轨迹条控件的父窗口，控件正在释放鼠标捕获。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE  
    lpnmh = (LPNMHDR) lParam;
```

参数

lpnmh: NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略来自这个通告消息返回值。

环境需求

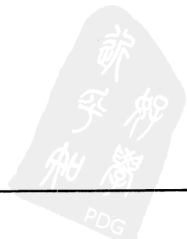
需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者，使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。



第27章 增减数控件

增减数控件 (up-down control) 是一对箭头按钮, 用户单击这两个按钮就可以使一个值进行自增或自减, 例如改变滚动位置或显示在伙伴控件中的数字。与增减数控件相关联的那个值被称为是当前位置。增减数控件经常与伙伴控件一同使用, 这个伙伴控件又称为伴随窗口。

27.1 关于增减数控件

对于用户来说, 增减数控件及其伴随窗口看起来就像一个控件一样。可以指定增减数控件自动地位于其伴随窗口的旁边, 并且可以在增减数控件的位置自动地设置伴随窗口的字幕。例如, 可结合使用一个增减数控件与一个编辑控件, 用来请求用户输入数据值。图27-1演示了一个以编辑控件作为其伴随窗口的增减数控件, 它们组合在一起有时又被称为是微调控件。



图27-1 增减数控件

没有伴随窗口的增减数控件在功能上就像一个简化的滚动控件一样。例如, 有时选项卡控件需要显示一个增减数控件, 用来让用户将更多的选项卡滚动到视图中。图27-2演示了一个选项卡控件右上角的增减数控件。

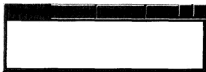


图 27-2

可以创建增减数控件, 并且按照几种方法来指定其伴随窗口。UPDOWN_CLASS值可以指定一个增减数控件的窗口类。可以在一个对话框模板中指定这个窗口类, 也可以在调用CreateWindowEx函数时指定这个窗口类。另一种方法就是使用CreateUpDownControl函数来创建增减数控件, 并且同时指定其伴随窗口、当前位置、最小位置与最大位置。

当通用控件动态链接库 (DLL) 被加载时, UPDOWN_CLASS窗口类将被注册。如果没有使用CreateUpDownControl函数来创建增减数控件, 那么必须确保通用控件动态链接库被加载, 为此, 可以使用InitCommonControls函数来检查这个通用控件动态链接库是否被加载。

CreateUpDownControl还可用来指定伴随窗口。如果没有使用这个函数创建增减数控件, 那么应该指定UDS_AUTOBUDDY窗口样式或使用UDM_SETBUDDY消息来设置伴随窗口。如果指定了UDS_AUTOBUDDY窗口样式, 那么增减数控件将自动地选择先前的Z轴次序窗口作为伴随窗口, 并且这个窗口必须是对话框模板中的先前控件。也可以使用UDM_SETBUDDY将一个特定的伴随窗口分配给增减数控件。如果需要确定增减数控件当前的伴随窗口, 则应该使用UDM_GETBUDDY消息。增减数控件及其伴随窗口必须具有相同的父窗口。

在增减数控件的当前位置发生改变时, 此控件将向父窗口发送UDN_DELTAPOS通告消息以及WM_VSCROLL消息或WM_HSCROLL消息。对于竖直的增减数控件 (没有使用UDS_HORZ样式), 它将发送WM_VSCROLL消息。对于水平增减数控件 (没有使用UDS_HORZ样式), 它

将发送UDS_HSCROLL消息。

27.1.1 关于增减数控件样式

利用窗口样式，可以控制增减数控件的性状，例如控制增减数控件与其伴随窗口的位置关系、是否为伴随窗口设置文本、是否处理向上箭头键与向下箭头键等等。

带有UDS_ALIGNLEFT样式或UDS_ALIGNRIGHT样式的增减数控件将与伴随窗口的左边或右边相对齐。同时，伴随窗口的宽度将减少，从而能够放置增减数控件。

在当前位置发生改变时，UDS_SETBUDDYINT样式的增减数控件将设置伴随窗口的字幕。除非指定了UDS_NOTHOUSANDS样式，否则控件将在数字字符串的每三个数字之间插入一个千分符。如果伴随窗口是一个列表框，那么将设置自己的当前选择，而不是设置字幕。

可以为增减数控件指定UDS_ARROWKEYS样式，从而提供键盘操作界面。如果指定了这个样式，这个控件就能够处理向上箭头键与向下箭头键。控件还将把伴随窗口作为自己的子类，从而在伴随窗口拥有焦点时，增减数控件可以处理这些键。

如果使用增减数控件进行水平滚动，就必须指定UDS_HORZ样式。这个样式将会导致控件的箭头指向左边和右边，而不是指向上方和下方。

缺省情况下，在用户试图增加当前位置的值使其大于最大值，或者减少当前位置的值使其小于最小值时，控件的当前位置不会改变。但如果使用了UDS_WRAP样式，这种行为将会发生改变，从而使得当前位置能够“绕”到位置值的另一个极端。例如，在当前值增加到超过增减数控件的上限时，这个值将绕回到下限值。

27.1.2 位置与加速度

在创建增减数控件之后，就可以利用消息改变此控件的当前位置、最小位置与最大位置。也可以改变用来在伴随窗口中显示当前位置的数字基数以及在单击向上箭头与向下箭头时当前位置改变的频率。

如果需要获取增减数控件的当前位置，那么就应该使用UDM_GETPOS消息。对于带有伴随窗口的增减数控件，当前位置就是伴随窗口字幕中的数字。由于字幕可能会发生改变（例如用户可能会设置编辑控件中的文本），因此增减数控件将会获取当前字幕并且相应地更新当前位置。

根据所使用的数字基数不同（可以是10进制也可以是16进制），伴随窗口字幕可以显示十进制字符串，也可以显示十六进制字符串。利用UDM_SETBASE消息可以设置所使用的数字基数，利用UDM_GETBASE消息可以获取正在被使用的数字基数。

UDM_SETPOS消息可用来设置伴随窗口的当前位置。需要说明的是，与滚动条不同，增减数控件在自己的向上箭头与向下箭头被单击时将自动地修改当前位置。因此，应用程序在处理WM_VSCROLL消息或WM_HSCROLL消息时不需要设置当前位置。

可以利用UDM_SETRANGE消息来改变增减数控件的最小位置与最大位置。其中，最大位置可能会小于最小位置，在这种情况下，单击向上箭头将会减少当前位置。从另一个角度来看，这时的向上就意味着向最大位置的方向运动。如果需要获取增减数控件的对象位置与最大位置，

应该使用UDM_GETRANGE消息。

通过设置增减数控件的加速度，就可以控制在用户按下某个箭头按钮时位置移动的频率。加速度被定义为UDACCEL数据结构的一个数组，其中每个数据结构定义了一个时间间隔以及在这个时间间隔内进行自增或自减的位置单位数。如果需要设置加速度，就可以使用UDM_SETACCEL消息。如果需要获取加速度信息，就应该使用UDM_GETACCEL消息。

27.1.3 缺省增减数控件消息处理

本节讨论由增减数控件所使用的标准窗口消息处理过程。

消 息	执行的动作
WM_CREATE	分配并初始化一个私有数据结构，并且保存此数据结构的地址作为窗口数据
WM_DESTROY	释放在处理WM_CREATE消息的过程中所分配的数据
WM_ENABLE	使窗口失效
WM_KEYDOWN	在处理向上箭头或向下箭头时，改变当前位置
WM_KEYUP	完成位置的改变
WM_LBUTTONDOWN	捕获鼠标。如果伴随窗口是一个编辑控件或列表框，那么将把焦点设置给伴随窗口；如果鼠标在向上按钮或向下按钮之上，那么改变当前位置并且设置计时器
WM_LBUTTONUP	完成位置的改变，并且在增减数控件捕获了鼠标时释放鼠标捕获。如果伴随窗口是一个编辑控件，那么将选中编辑控件中的所有文本
WM_PAINT	绘制增减数控件。如果wParam参数不是NULL，那么控件将假设这个值是一个HDC，并且利用这个设备上下文进行绘制
WM_TIMER	如果鼠标正在某个按钮上被按下，并且经过了足够的时间间隔，那么将改变当前位置

27.2 在Internet Explorer中更新增减数控件

Microsoft Internet Explorer中的增减数控件支持下列新特性。

完全的32位范围

现在，增减数控件已经可以支持完全的32位范围了。新增加的UDM_SETRANGE32消息与UDM_GETRANGE32消息可以用来是实现此特性。增减数控件使用带符号的整数作为自己的范围，因此为了能够使用完全32位范围，必须考虑将范围值设置在-0x7FFFFFFF到+0x7FFFFFFF之间。

27.3 增减数控件样式

在创建增减数控件时可以使用下列样式：

UDS_ALIGNLEFT	这个样式将把增减数控件放置在伴随窗口的左边。伴随窗口被移动到右边，并且它的宽度将减小，从而能够放置增减数控件。
UDS_ALIGNRIGHT	这个样式将把增减数控件放置在伴随窗口的右边。并且伴

UDS_ARROWKEYS	随窗口的宽度将减小，从而能够放置增减数控件。
UDS_AUTOBUDDY	这个样式将导致增减数控件在向上箭头键与向下箭头键被按下时自增或自减自己的当前位置。
UDS_HORZ	这个样式将自动地从先前的Z轴次序窗口中选择一个窗口作为增减数控件的伴随窗口。
UDS_HOTTRACK	这个样式将使增减数控件的箭头指向左边与右边，而不是向上与向下。
UDS_NOTHOUSANDS	这个样式将导致控件显示出“热跟踪”的行为。也就是说，当鼠标指针移动到控件的向上箭头与向下箭头时，控件将对它们进行高亮显示。这个样式需要系统运行于Windows 98或Windows 2000。如果系统运行于Windows 95或Windows NT 4.0，那么这个标志位将被忽略。可以调用SystemParametersInfo函数，来判断是否打开了热跟踪功能。
UDS_SETBUDDYINT	这个样式将使得控件不会在每十进制数字之间插入一个千分符。
UDS_WRAP	这个样式将导致增减数控件增量位置发生改变时设置伴随窗口的文本（使用WM_SETTEXT消息）。在这个文本中包含有被格式化为十进制字符串或十六进制字符串的位置信息。
	如果当前位置在自增或自减时越过了范围的结束位置或范围的开始位置，那么这个样式将使得位置“绕”到另一个极端。

27.4 增减数控件参考

27.4.1 增减数控件函数

CreateUpDownControl

创建增减数控件。

```
HWND CreateUpDownControl(
    DWORD dwStyle,
    int x,
    int y,
    int cx,
    int cy,
    HWND hParent,
    int nID,
```

```

HINSTANCE hInst
HWND hBuddy ,
int nUpper ,
int nLower ,
int nPos
);

```

参数

dwStyle: 控件的窗口样式。这个参数应该包括WS_CHILD样式、WS_BORDER样式、WS_VISIBLE样式, 并且可以包括任何增减数控件所特有的窗口样式。

x: 增减数控件左上角的水平客户区坐标。

y: 增减数控件左上角的竖直客户区坐标。

cx: 增减数控件的宽度, 单位为像素。

cy: 增减数控件的高度, 单位为像素。

hParent: 增减数控件父窗口的句柄。

nID: 增减数控件的标识符。

hInst: 创建增减数控件的应用程序模块实例句柄。

hBuddy: 与增减数控件相关联的窗口的句柄。如果这个参数为NULL, 那么表示控件没有任何伴随窗口。

nUpper: 增减数控件位置范围的上限。

nLower: 增减数控件位置范围的下限。

nPos: 增减数控件的位置。

返回值

如果操作成功, 则返回增减数控件的窗口句柄值。否则, 返回NULL。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

导入库: comctl32.lib。

27.4.2 增减数控件消息

UDM_GETACCEL

获取增减数控件的加速度信息。

```

UDM_GETACCEL
wParam = (WPARAM) cAccels;
lParam = (LPARAM) (LPUDACCEL) paAccels;

```

参数

cAccels: 由paAccels所指定的数组元素编号。

paAccels: UDACCEL数据结构数组的地址, 这个数组将用来获取加速度信息。

返回值

所得到的返回值是加速度数据结构所获取的值。

如果cAccels参数为0, 并且paAccels参数为NULL, 那么返回值就是当前为控件所设置的加速度数目。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

UDM_SETACCEL。

UDM_GETBASE

获取增减数控件的当前数字基数(10进制还是16进制)。

UDM_GETBASE

wParam = 0;

lParam = 0;

返回值

返回当前的数字基数值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

UDM_GETBUDDY

获取当前伴随窗口的句柄。

UDM_GETBUDDY

wParam = 0;

lParam = 0;

返回值

返回当前伴随窗口的句柄。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

UDM_GETPOS

获取增减数控件当前位置的16位精度值。

```
UDM_GETPOS  
wParam = 0;  
lParam = 0;
```

返回值

如果操作成功, 则返回值的高位字部分被设置为0, 而返回值的低位字部分则被设置为控件的当前位置。如果出现错误, 那么返回值的高位字部分将被设置为非零值。

说明

在处理这个消息时, 增减数控件将根据伴随窗口中的字幕来更新增减数控件的当前位置。如果不存在任何伴随窗口, 或者伴随窗口的字幕中指定了非法的或超越范围的值, 增减数控件将返回一个错误信息。

如果为UDM_SETRANGE32样式的增减数控件打开了32位值功能, 那么这个消息将只返回当前位置的低16位值。如果需要获取完全的32位值, 那么应该使用UDM_GETPOS32。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

UDM_GETRANGE、UDM_GETRANGE32、UDM_SETPOS、UDM_SETRANGE32。

UDM_GETRANGE

获取增减数控件的最小位置与最大位置(即范围)。

```
UDM_GETRANGE  
wParam = 0;  
lParam = 0;
```

返回值

返回一个包含有最小位置与最大位置的32位值。其中, 返回值的低位字部分放置了控件的最大位置值, 高位字部分放置了控件的最小位置值。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

UDM_GETRANGE32

获取增减数控件的32位范围值。

```
UDM_GETRANGE32
    wParam = (WPARAM)(LPINT) pLow;
    lParam = (LPARAM)(LPINT) pHigh;
```

参数

pLow: 一个带符号整数的地址, 这个整数将接收增减数控件范围值的下限。此参数可以是NULL。

pHigh: 一个带符号整数的地址, 这个整数将接收增减数控件范围值的上限。此参数可以是NULL。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

UDM_GETUNICODEFORMAT

获取控件的UNICODE字符格式标志。

```
UDM_GETUNICODEFORMAT
    wParam = 0;
    lParam = 0;
```

返回值

返回控件的UNICODE格式标志。如果返回值非零, 那么表示控件正在使用UNICODE字符; 如果返回值为0, 那么表示控件正在使用ANSI字符。

说明

关于这个消息的更多讨论, 请参见CCM_GETUNICODEFORMAT。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000: 需要使用Windows NT 4.0或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

UDM_SETUNICODEFORMAT。

UDM_SETACCEL

设置增减数控件的加速度。

```
UDM_SETACCEL
    wParam = (WPARAM) nAccels;
    lParam = (LPARAM) (LPUDACCEL) aAccels;
```

参数

nAccels: 由aAccels所指定的UDACCEL数据结构数组的元素数。

aAccels: UDACCEL数据结构数组的地址, 在这个数据结构中包含有加速度信息。在这个数组中的元素应该根据nSec数据成员按照上升的顺序进行存储。

返回值

若操作成功, 则返回TRUE; 否则, 返回FALSE。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

UDM_GETACCEL。

UDM_SETBASE

为增减数控件设置数字基数。数字基数值能够决定伴随窗口是按照十进制数的形式显示数值还是按照十六进制数的形式显示数值。其中, 十六进制数中是不带符号的, 而十进制数可以带符号。

```
UDM_SETBASE
    wParam = (WPARAM) nBase;
    lParam = 0;
```

参数

nBase: 将要为增减数控件所设置的数字基数值。对于十进制数, 这个参数为 10; 对于十六进制数, 这个参数为 16。

返回值

所得到返回值是先前数字基数值。如果指定了非法的数字基数, 则返回0。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

UDM_SETBUDDY

为增减数控件设置伴随窗口。

```
UDM_SETBUDDY  
    wParam = (WPARAM)(HWND) hwndBuddy;  
    lParam = 0;
```

参数

hwndBuddy：新伴随窗口的句柄。

返回值

所得到的返回值是先前伴随窗口的句柄。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

UDM_SETPOS

为增减数控件设置16位精度的当前位置值。

```
UDM_SETPOS  
    wParam = 0;  
    lParam = (LPARAM) MAKELONG ((short) nPos, 0 );
```

参数

nPos：增减数控件的新位置。如果这个参数超出了控件的指定范围，那么nPos将被设置为最靠近的那个合法值。

返回值

返回先前位置值。

说明

这个消息只能支持16位位置。如果为带有UDM_SETRANGE32样式的增减数控件打开了32位值功能，那么就应该使用UDM_SETPOS32。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

参见

UDM_GETRANGE、UDM_GETPOS。

UDM_SETRANGE

为增减数控件设置最小位置与最大位置（即范围）。

```
UDM_SETRANGE  
    wParam = 0;  
    lParam = (LPARAM) MAKELONG((short) nUpper, (short)  
    nLower);
```

参数

nUpper和**nLower**：增减数控件的最大位置与最小位置。这两个位置都不能大于UD_MAXVAL值，也不能小于UD_MINVAL值。此外，这两个位置值之差不能超过UD_MAXVAL。

返回值

没有返回值。

说明

最大位置可能会小于最小位置。这时，单击向上箭头按钮将会使得当前位置向靠近最大位置的方向移动，而单击向下箭头按钮将会使得当前位置向靠近最小位置的方向移动。

环境需求

Windows NT/2000：需要使用Windows NT 3.51或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：需要使用Windows CE 1.0或更新版本。

头文件：在commctrl.h中进行了声明。

UDM_SETRANGE32

设置增减数控件的32位范围。

```
UDM_SETRANGE32  
    wParam = (WPARAM)(int) iLow;  
    lParam = (LPARAM)(int) iHigh;
```

参数

iLow：用来表示增减数控件范围新下限的带符号整型值。

iHigh：用来表示增减数控件范围新上限的带符号整型值。

返回值

这个消息的返回值没有被使用。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000：需要使用Windows 2000（或者，使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0）。

Windows 95/98：需要使用Windows 98（或者，使用带有Internet Explorer 4.0或更新版本的Windows 95）。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

UDM_SETUNICODEFORMAT

设置控件的UNICODE字符格式标志。这个消息允许在运行时动态地改变由控件所使用的字符，而不必重新创建控件。

```
UDM_SETUNICODEFORMAT
wParam = (WPARAM)(BOOL) fUnicode;
lParam = 0;
```

参数

fUnicode：判断控件所使用的字符集。如果这个值为非零，那么表示控件将使用UNICODE字符；如果这个值为0，那么表示控件将使用ANSI字符。

返回值

返回控件的先前UNICODE格式标志。

说明

关于这个消息的更多信息，请参见CCM_SETUNICODEFORMAT的说明。

环境需求

需要使用动态链接库Comctl32.dll的4.00版本或更新版本。

Windows NT/2000：需要使用Windows NT 4.0或更新版本。

Windows 95/98：需要使用Windows 95或更新版本。

Windows CE：不支持Windows CE。

头文件：在commctrl.h中进行了声明。

参见

UDM_GETUNICODEFORMAT。

27.4.3 增减数控件通告消息

NM_RELEASEDCAPTURE（增减数控件）

这个通告消息用来告知增减数控件的父窗口，控件正在释放鼠标捕获。这个通告消息是以WM_NOTIFY消息的形式进行发送的。

```
NM_RELEASEDCAPTURE
lpmh = (LPNMHDR) lParam;
```

参数

lpmh：NMHDR数据结构的地址，在这个数据结构中包含有关于本通告消息的更多信息。

返回值

控件将忽略来自这个通告消息的返回值。

环境需求

需要使用动态链接库Comctl32.dll的4.71版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 4.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

UDN_DELTAPOS

当增减数控件的位置将要发生改变时, 操作系统将向增减数控件的父窗口发送这个通告消息。例如, 在用户按下控件的向上箭头或向下箭头时, 就可能请求改变位置。UDN_DELTAPOS通告消息是以WM_NOTIFY消息的形式进行发送的。

```
UDN_DELTAPOS
lpnmud = (LPNMUPDOWN) lParam;
```

参数

lpnmud: NMUPDOWN数据结构的地址, 在这个数据结构中包含有关于位置改变的信息。

在此数据结构的iPos数据成员中包含有控件的当前位置信息, 在iDelta数据成员中包含有将要改变的位置增量(带符号整数)。如果用户单击了向上按钮, 那么这个数据成员就是一个正值。单击了向下按钮, 那么这个数据成员就是一个负值。

返回值

如果需要阻止控件的位置发生改变, 则返回非零值; 如果允许位置发生改变, 则返回0。

说明

UDN_DELTAPOS通告消息在WM_VSCROLL消息或WM_HSCROLL消息之前发送, 而后面两个才真正改变控件的位置。这样做的好处是允许对将要改变的位置进行检查、许可、修改或拒绝。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

参见

WM_COMMAND。

27.4.4 增减数控件数据结构

NMUPDOWN

在这个数据结构中包含有增减数控件通告消息所特有的信息, 它正在逐渐取代NM_UPDOWN数据结构。

```
typedef struct _NM_UPDOWN {
    NMHDR hdr;
    int iPos;
    int iDelta;
}NMUPDOWN, FAR *LPNMUPDOWN;
```

成员

hdr: 这个数据成员是一个NMHDR数据结构, 其中包含有关于本通告消息的更多信息。

iPos: 这个数据成员是一个带符号整数, 用来表示增减数控件的当前位置。

iDelta: 这个数据成员是一个带符号整数, 来表示增减数控件位置将要改变的增量。

环境需求

需要使用动态链接库Comctl32.dll的4.70版本或更新版本。

Windows NT/2000: 需要使用Windows 2000 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows NT 4.0)。

Windows 95/98: 需要使用Windows 98 (或者, 使用带有Internet Explorer 3.0或更新版本的Windows 95)。

Windows CE: 不支持Windows CE。

头文件: 在commctrl.h中进行了声明。

参见

UDN_DELTAPOS、WM_NOTIFY。

UDACCEL

在这个数据结构中包含有增减数控件的加速度信息。

```
typedef struct {
    UINT nSec;
    UINT nInc;
}UDACCEL, FAR *LPUDACCEL;
```

成员

nSec: 这个数据成员用来表示在使用由nInc所指定的位置变化量之前, 应该等待的时间, 单位为秒。

nInc: 这个数据成员来表示在等待了由nSec所指定的秒数之后, 当前位置的变化量。

环境需求

Windows NT/2000: 需要使用Windows NT 3.51或更新版本。

Windows 95/98: 需要使用Windows 95或更新版本。

Windows CE: 需要使用Windows CE 1.0或更新版本。

头文件: 在commctrl.h中进行了声明。

索引A 按技术分组编程元素

本书的索引使得从“Win32开发人员参考库”各卷中查找信息变得更为容易。笔者没有将各个纲领性的编程元素混乱地放到目录中（这样使用目录会导致不必要的跳转），而是将其编成索引放在每卷的后面。通过使用这些索引，读者可以方便地找到自己感兴趣的元素——并且可以快速且容易地知道该元素在该卷的哪个地方。

同时，为了使读者与Microsoft 技术同步，笔者创建了一个基于Web的文档，并且将Microsoft的某些技术映射到了该文档中。在该文档中，读者可以得到更多的信息。以下链接可使读者获取最新Microsoft技术的索引：www.iseminger.com/winprs/technologies。

如果读者的建议有助于这部分的改进，别忘了将反馈信息寄给笔者。笔者不能保证一定答复，但一定会仔细阅读。如果您的建议对其他人的帮助，笔者会将其融合到将来的卷中。

通用API参考 62

通用API函数 62

GetEffectiveClientRect

GetMUILanguage

InitCommonControls

InitCommonControlsEx

InitMUILanguage

ShowHideMenuCtl

通用API消息 65

CCM_GETUNICODEFORMAT

CCM_GETVERSION

CCM_SETUNICODEFORMAT

CCM_SETVERSION

WM_NOTIFY

WM_NOTIFYFORMAT

通用API宏 71

FORWARD_WM_NOTIFY

HANDLE_WM_NOTIFY

INDEXTOSTATEIMAGEMASK

通用API通告消息 73

NM_CHAR

NM_CLICK

NM_DBLCLK

NM_HOVER

NM_KEYDOWN

NM_KILLFOCUS

NM_NCHITTEST

NM_OUTOFMEMORY

NM_RCLICK

NM_RDBLCLK

NM_RELEASEDCAPTURE

NM_RETURN

NM_SETCURSOR

NM_SETFOCUS

NM_TOOLTIPSCREATED

通用API数据结构 80

COLORSCHEME

INITCOMMONCONTROLSEX

NMCHAR

NMHDR

NMKEY

NM_MOUSE

NMOBJECTNOTIFY

NM_TOOLTIPSCREATED

定制绘图参考 90

- 定制绘图通告消息 90
- NM_CUSTOMDRAW
- 定制绘图数据结构 92
- NMCUSTOMDRAW

动画控件参考 97

- 动画控件消息 97
- ACM_OPEN
- ACM_PLAY
- ACM_STOP
- 动画控件宏 99
- Animate_Close
- Animate_Create
- Animate_Open
- Animate_OpenEx
- Animate_Play
- Animate_Seek
- Animate_Stop

动画控件通告消息 104

- ACN_START
- ACN_STOP

扩展组合框控件参考 113

- 扩展组合框控件消息 113
- CBEM_DELETEITEM
- CIBEM_GETCOMBOCONTROL
- CBEM_GETEDITCONTROL
- CBEM_GETEXTENDEDSTYLE
- CBEM_GETIMAGELIST
- CBEM_GETITEM
- CBEM_GETUNICODEFORMAT
- CBEM_HASEDITCHANGED
- CBEM_INSERTITEM
- CBEM_SETEXTENDEDSTYLE
- CBEM_SETIMAGELIST
- CBEM_SETITEM

CBEM_SETUNICODEFORMAT

- 扩展组合框控件通告消息 120
- CBEN_BEGINEDIT
- CBEN_DELETEITEM
- CBEN_DRAGBEGIN
- CBEN_ENDEDIT
- CBEN_GETDISPINFO
- CBEN_INSERTITEM
- NM_SETCURSOR (扩展组合框)
- 扩展组合框控件数据结构 123
- COMBOBOXEXITEM
- NMCBEENDEDIT
- NMCBEDRAGBEGIN
- NMCOMBOBOXEX

日期与时间检出器参考 155

- 日期与时间检出器控件消息 155
- DTM_GETMCCOLOR
- DTM_GETMCFONT
- DTM_GETMONTHCAL
- DTM_GETRANGE
- DTM_GETSYSTEMTIME
- DTM_SETFORMAT
- DTM_SETMCCOLOR
- DTM_SETMCFONT
- DTM_SETRANGE
- DTM_SETSYSTEMTIME

日期与时间检出器控件宏 162

- DateTime_GetMonthCal
- DateTime_GetMonthCalColor
- DateTime_GetMonthCalFont
- DateTime_GetRange
- DateTime_GetSystemtime
- DateTime_SetFormat
- DateTime_SetMonthCalColor
- DateTime_SetMonthCalFont
- DateTime_SetRange

- DateTime_SetSystemtime
- 日期与时间检出器控件通告消息 168
 - DTN_CLOSEUP
 - DTN_DATETIMECHANGE
 - DTN_DROPDOWN
 - DTN_FORMAT
 - DTN_FORMATQUERY
 - DTN_USERSTRING
 - DTN_WMKEYDOWN
 - NM_KILLFOCUS (date time)
 - NM_SETFOCUS (date time)
- 日期与时间检出器控件数据结构 174
 - NMDATETIMECHANGE
 - NMDATETIMEFORMAT
 - NMDATETIMEFORMATQUERY
 - NMDATETIMESTRING
 - NMDATETIMEWMKEYDOWN
- 拖放列表框参考 179
 - 拖放列表框函数 179
 - DrawInsert
 - LBItemFromPt
 - MakeDragList
 - 拖放列表框通告消息 180
 - DL_BEGINDRAG
 - DL_CANCELDRAG
 - DL_DRAGGING
 - DL_DROPPED
 - 拖放列表框数据结构 183
 - DRAGLISTINFO
- 平面滚动条参考 186
 - 平面滚动条函数 186
 - InitializeFlatSB
 - FlatSB_EnableScrollBar
 - FlatSB_GetScrollInfo
 - FlatSB_GetScrollPos
 - FlatSB_GetScrollProp
 - FlatSB_GetScrollRange
 - FlatSB_SetScrollInfo
 - FlatSB_SetScrollPos
 - FlatSB_SetScrollProp
 - FlatSB_SetScrollRange
 - FlatSB_ShowScrollBar
 - UninitializeFlatSB
- 头标控件参考 205
 - 头标控件消息 205
 - HDM_CLEARFILTER
 - HDM_CREATEDRAGIMAGE
 - HDM_DELETEITEM
 - HDM_EDITFILTER
 - HDM_GETBITMAPMARGIN
 - HDM_GETIMAGELIST
 - HDM_GETITEM
 - HDM_GETITEMCOUNT
 - HDM_GETITEMRECT
 - HDM_GETORDERARRAY
 - HDM_GETUNICODEFORMAT
 - HDM_HITTEST
 - HDM_INSERTITEM
 - HDM_LAYOUT
 - HDM_ORDERTOINDEX
 - HDM_SETBITMAPMARGIN
 - HDM_SETFILTERCHANGETIMEOUT
 - HDM_SETHOTDIVIDER
 - HDM_SETIMAGELIST
 - HDM_SETITEM
 - HDM_SETORDERARRAY
 - HDM_SETUNICODEFORMAT
 - 头标控件宏 217
 - Header_ClearFilter
 - Header_CreateDragImage
 - Header_DeleteItem

- Header_EditFilter
- Header_GetBitmapMargin
- Header_GetImageList
- Header_GetItem
- Header_GetItemCount
- Header_GetItemRect
- Header_GetOrderArray
- Header_GetUnicodeFormat
- Header_InsertItem
- Header_Layout
- Header_OrderToIndex
- Header_SetBitmapMargin
- Header_SetFilterChangeTimeout
- Header_SetHotDivider
- Header_SetImageList
- Header_SetItem
- Header_SetOrderArray
- Header_SetUnicodeFormat
- 头标控件通告消息 231
- HDN_BEGINDRAG
- HDN_BEGINTRACK
- HDN_DIVIDERDBLCLICK
- HDN_ENDDRAG
- HDN_ENDTRACK
- HDN_FILTERBTNCLICK
- HDN_FILTERCHANGE
- HDN_GETDISPINFO
- HDN_ITEMCHANGED
- HDN_ITEMCHANGING
- HDN_ITEMCLICK
- HDN_ITEMDBLCLICK
- HDN_TRACK
- NM_CUSTOMDRAW (头标)
- NM_RCLICK (头标)
- NM_RELEASEDCAPTURE (头标)
- 头标控件数据结构 239
- HDHITTESTINFO
- HDITEM
- HDLAYOUT
- HDTEXTFILTER数据结构
- NMHDDISPINFO
- NMHDFILTERBTNCLICK数据结构
- NMHEADER
- 热键控件参考 251
- 热键控件消息 251
- HKM_GETHOTKEY
- HKM_SETHOTKEY
- HKM_SETRULES
- IP 地址控件参考 254
- IP 地址控件消息 254
- IPM_CLEARADDRESS
- IPM_GETADDRESS
- IPM_ISBLANK
- IPM_SETADDRESS
- IPM_SETFOCUS
- IPM_SETRANGE
- IP 地址控件通告消息 257
- IPN_FIELDCHANGED
- IP 地址控件宏 258
- FIRST_IPADDRESS
- FOURTH_IPADDRESS
- MAKEIPADDRESS
- MAKEIPRANGE
- SECOND_IPADDRESS
- THIRD_IPADDRESS
- IP 地址控件数据结构 261
- NMIPADDRESS
- 月历控件参考 269
- 月历控件消息 269
- MCM_GETCOLOR
- MCM_GETCURSEL

- MCM_GETFIRSTDAYOFWEEK
- MCM_GETMAXSELCOUNT
- MCM_GETMAXTODAYWIDTH
- MCM_GETMINREQRECT
- MCM_GETMONTHDELTA
- MCM_GETMONTHRANGE
- MCM_GETRANGE
- MCM_GETSELRange
- MCM_GETTODAY
- MCM_GETUNICODEFORMAT
- MCM_HITTEST
- MCM_SETCOLOR
- MCM_SETCURSEL
- MCM_SETDAYSTATE
- MCM_SETFIRSTDAYOFWEEK
- MCM_SETMAXSELCOUNT
- MCM_SETMONTHDELTA
- MCM_SETRANGE
- MCM_SETSELRange
- MCM_SETTODAY
- MCM_SETUNICODEFORMAT
- 月历控件宏 285
 - MonthCal_GetColor
 - MonthCal_GetCurSel
 - MonthCal_GetFirstDayOfWeek
 - MonthCal_GetMaxSelCount
 - MonthCal_GetMaxTodayWidth
 - MonthCal_GetMinReqRect
 - MonthCal_GetMonthDelta
 - MonthCal_GetMonthRange
 - MonthCal_GetRange
 - MonthCal_GetSelRange
 - MonthCal_GetToday
 - MonthCal_GetUnicodeFormat
 - MonthCal_HitTest
 - MonthCal_SetColor
 - MonthCal_SetCurSel
 - MonthCal_SetDayState
 - MonthCal_SetFirstDayOfWeek
 - MonthCal_SetMaxSelCount
 - MonthCal_SetMonthDelta
 - MonthCal_SetRange
 - MonthCal_SetSelRange
 - MonthCal_SetToday
 - MonthCal_SetUnicodeFormat
- 月历控件通告消息 300
 - MCN_GETDAYSTATE
 - MCN_SELCHANGE
 - MCN_SELECT
 - NM_RELEASEDCAPTURE (monthcal)
- 月历控件数据结构 303
 - MCHITTESTINFO
 - NMIDAYSTATE
 - NMSELCHANGE
- 月历控件数据类型 305
 - MONTHDAYSTATE
- Pager 控件参考 308
 - Pager 控件消息 308
 - PGM_FORWARDMOUSE
 - PGM_GETBKCOLOR
 - PGM_GETBORDER
 - PGM_GETBUTTONSIZE
 - PGM_GETBUTTONSTATE
 - PGM_GETDROPTARGET
 - PGM_GETPOS
 - PGM_RECALCSIZE
 - PGM_SETBKCOLOR
 - PGM_SETBORDER
 - PGM_SETBUTTONSIZE
 - PGM_SETCHILD
 - PGM_SETPOS
- Pager 控件宏 315
 - Pager_ForwardMouse

- Pager_GetBkColor
- Pager_GetBorder
- Pager_GetButtonSize
- Pager_GetButtonState
- Pager_GetDropTarget
- Pager_GetPos
- Pager_RecalcSize
- Pager_SetBkColor
- Pager_SetBorder
- Pager_SetButtonSize
- Pager_SetChild
- Pager_SetPos
- Pager 控件通告消息 323
 - NM_RELEASEDCAPTURE (pager)
 - PGN_CALCSIZE
 - PGN_SCROLL
- Pager 控件数据结构 324
 - NMPGCALCSIZE
 - NMPGSCROLL
- 进度条控件参考 330
 - 进度条控件消息 330
 - PBM_DELTAPOS
 - PBM_GETPOS
 - PBM_GETRANGE
 - PBM_SETBARCOLOR
 - PBM_SETBKCOLOR
 - PBM_SETPOS
 - PBM_SETRANGE
 - PBM_SETRANGE32
 - PBM_SETSTEP
 - PBM_STEPIT
 - 进度条控件数据结构 335
 - PBRANGE
- 属性页参考 346
 - 属性页函数 346
 - AddPropSheetPageProc
 - CreatePropertySheetPage
 - DestroyPropertySheetPage
 - ExtensionPropSheetPageProc
 - PropertySheet
 - PropSheetPageProc
 - PropSheetPageProc
 - 属性页消息 351
 - PSM_ADDPAGE
 - PSM_APPLY
 - PSM_CANCELTOCLOSE
 - PSM_CHANGED
 - PSM_GETCURRENTPAGEHWND
 - PSM_GETTABCONTROL
 - PSM_HWNDTOINDEX
 - PSM_IDTOINDEX
 - PSM_INDEXTOHWND
 - PSM_INDEXTOID
 - PSM_INDEXTOPAGE
 - PSM_INSERTPAGE
 - PSM_ISDIALOGMESSAGE
 - PSM_PAGETOINDEX
 - PSM_PRESSBUTTON
 - PSM_QUERYSIBLINGS
 - PSM_REBOOTSYSTEM
 - PSM_REMOVEPAGE
 - PSM_RESTARTWINDOWS
 - PSM_SETCURSEL
 - PSM_SETCURSELID
 - PSM_SETFINISHTEXT
 - PSM_SETHEADERSUBTITLE
 - PSM_SETHEADERTITLE
 - PSM_SETTITLE
 - PSM_SETWIZBUTTONS
 - PSM_UNCHANGED
 - 属性页宏 366
 - PropSheet_AddPage

PropSheet_Apply
 PropSheet_CancelToClose
 PropSheet_Changed
 PropSheet_GetCurrentPageHwnd
 PropSheet_GetTabControl
 PropSheet_HwndToIndex
 PropSheet_IdToIndex
 PropSheet_IndexToHwnd
 PropSheet_IndexToId
 PropSheet_IndexToPage
 PropSheet_InsertPage
 PropSheet_IsDialogMessage
 PropSheet_PageToIndex
 PropSheet_PressButton
 PropSheet_QuerySiblings
 PropSheet_RebootSystem
 PropSheet_RemovePage
 PropSheet_RestartWindows
 PropSheet_SetCurSel
 PropSheet_SetCurSelById
 PropSheet_SetFinishText
 PropSheet_SetHeaderSubTitle
 PropSheet_SetHeaderTitle
 PropSheet_SetTitle
 PropSheet_SetWizButtons
 PropSheet_UnChanged
 属性页通告消息 383
 PSN_APPLY
 PSN_GETOBJECT
 PSN_HELP
 PSN_KILLACTIVE
 PSN_QUERYCANCEL
 PSN_QUERYINITIALFOCUS
 PSN_RESET
 PSN_SETACTIVE
 PSN_TRANSLATEACCELERATOR
 PSN_WIZBACK

PSN_WIZFINISH
 PSN_WIZNEXT
 属性页数据结构 392
 PROPSHEETHEADER
 PROPSHEETPAGE
 PSHNOTIFY

Rebar控件参考 405

Rebar 控件消息 405
 RB_BEGINDRAG
 RB_DELETEBAND
 RB_DRAGMOVE
 RB_ENDDRAG
 RB_GETBANDBORDERS
 RB_GETBANDCOUNT
 RB_GETBANDINFO
 RB_GETBARHEIGHT
 RB_GETBARINFO
 RB_GETBKCOLOR
 RB_GETCOLORSCHEME
 RB_GETDROPTARGET
 RB_GETPALETTE
 RB_GETRECT
 RB_GETROWCOUNT
 RB_GETROWHEIGHT
 RB_GETTEXTCOLOR
 RB_GETTOOLTIPS
 RB_GETUNICODEFORMAT
 RB_HITTEST
 RB_IDTOINDEX
 RB_INSERTBAND
 RB_MAXIMIZEBAND
 RB_MINIMIZEBAND
 RB_MOVEBAND
 RB_PUSHCHEVRON
 RB_SETBANDINFO
 RB_SETBARINFO

RB_SETBKCOLOR
 RB_SETCOLORSCHEME
 RB_SETPALETTE
 RB_SETPARENT
 RB_SETTEXTCOLOR
 RB_SETTOOLTIPS
 RB_SETUNICODEFORMAT
 RB_SHOWBAND
 RB_SIZETOEXT

Rebar控件通告消息 425

NM_CUSTOMDRAW (rebar)
 NM_NCHITTEST (rebar)
 NM_RELEASEDCAPTURE (rebar)
 RBN_AUTOSIZE
 RBN_BEGINDRAG
 RBN_CHEVRONPUSHED
 RBN_CHILDSIZE
 RBN_DELETEDBAND
 RBN_DELETINGBAND
 RBN_ENDDRAG
 RBN_GETOBJECT
 RBN_HEIGHTCHANGE
 RBN_LAYOUTCHANGED

Rebar控件数据结构 433

NMRBAUTOSIZE
 NMREBAR
 NMREBARCHEVRON
 NMREBARCHILDSIZE
 RBHITTESTINFO
 REBARBANDINFO
 REBARINFO

状态条参考 445

状态条函数 445

CreateStatusWindow
 DrawStatusText
 MenuHelp

状态条消息 448

SB_GETBORDERS
 SB_GETICON
 SB_GETPARTS
 SB_GETRECT
 SB_GETTEXT
 SB_GETTEXTLENGTH
 SB_GETTIPTTEXT
 SB_GETUNICODEFORMAT
 SB_ISSIMPLE
 SB_SETBKCOLOR
 SB_SETICON
 SB_SETMINHEIGHT
 SB_SETPARTS
 SB_SETTEXT
 SB_SETTIPTTEXT
 SB_SETUNICODEFORMAT
 SB_SIMPLE

状态条通告消息 458

NM_CLICK (状态条)
 NM_DBLCLK (状态条)
 NM_RCLICK (状态条)
 NM_RDBLCLK (状态条)
 SBN_SIMPLEMODECHANGE

选项卡控件参考 476

选项卡控件消息 476

TCM_ADJUSTRECT
 TCM_DELETEALLITEMS
 TCM_DELETEITEM
 TCM_DESELECTALL
 TCM_GETCURFOCUS
 TCM_GETCURREL
 TCM_GETEXTENDEDSTYLE
 TCM_GETIMAGELIST
 TCM_GETITEM
 TCM_GETITEMCOUNT

- TCM_GETITEMRECT
- TCM_GETROWCOUNT
- TCM_GETTOOLTIPS
- TCM_GETUNICODEFORMAT
- TCM_HIGHLIGHTITEM
- TCM_HITTEST
- TCM_INSERTITEM
- TCM_REMOVEIMAGE
- TCM_SETCURFOCUS
- TCM_SETCURSEL
- TCM_SETEXTENDEDSTYLE
- TCM_SETIMAGELIST
- TCM_SETITEM
- TCM_SETITEMEXTRA
- TCM_SETITEMSIZE
- TCM_SETMINTABWIDTH
- TCM_SETPADDING
- TCM_SETTOOLTIPS
- TCM_SETUNICODEFORMAT
- 选项卡控件宏 491
 - TabCtrl_AdjustRect
 - TabCtrl_DeleteAllItems
 - TabCtrl_DeleteItem
 - TabCtrl_DeselectAll
 - TabCtrl_GetCurFocus
 - TabCtrl_GetCurSel
 - TabCtrl_GetExtendedStyle
 - TabCtrl_GetImageList
 - TabCtrl_GetItem
 - TabCtrl_GetItemCount
 - TabCtrl_GetItemRect
 - TabCtrl_GetRowCount
 - TabCtrl_GetToolTips
 - TabCtrl_GetUnicodeFormat
 - TabCtrl_HighlightItem
 - TabCtrl_HitTest
 - TabCtrl_InsertItem
 - TabCtrl_RemoveImage
 - TabCtrl_SetCurFocus
 - TabCtrl_SetCurSel
 - TabCtrl_SetExtendedStyle
 - TabCtrl_SetImageList
 - TabCtrl_SetItem
 - TabCtrl_SetItemExtra
 - TabCtrl_SetItemSize
 - TabCtrl_SetMinTabWidth
 - TabCtrl_SetPadding
 - TabCtrl_SetToolTips
 - TabCtrl_SetUnicodeFormat
- 选项卡控件通告消息 507
 - NM_CLICK (选项卡)
 - NM_RCLICK (选项卡)
 - NM_RELEASEDCAPTURE (选项卡)
 - TCN_FOCUSCHANGE
 - TCN_GETOBJECT
 - TCN_KEYDOWN
 - TCN_SELCHANGE
 - TCN_SELCHANGING
- 选项卡控件数据结构 510
 - NMTCKEYDOWN
 - TCHITTESTINFO
 - TCITEM
 - TCITEMHEADER
- 工具提示控件参考 530
 - 工具提示控件消息 530
 - TTM_ACTIVATE
 - TTM_ADDTOOL
 - TTM_ADJUSTRECT
 - TTM_DELTOOL
 - TTM_ENUMTOOLS
 - TTM_GETBUBBLESIZE
 - TTM_GETCURRENTTOOL
 - TTM_GETDELAYTIME

- TTM_GETMARGIN
- TTM_GETMAXTIPWIDTH
- TTM_GETTEXT
- TTM_GETTIPBKCOLOR
- TTM_GETTOOLCOUNT
- TTM_GETTOOLINFO
- TTM_HITTEST
- TTM_NEWTOOLRECT
- TTM_POP
- TTM_RELAYEVENT
- TTM_SETDELAYTIME
- TTM_SETMARGIN
- TTM_SETMAXTIPWIDTH
- TTM SETTIPBKCOLOR
- TTM SETTIPTEXTCOLOR
- TTM SETTITLE
- TTM SETTOOLINFO
- TTM_TRACKACTIVATE
- TTM_TRACKPOSITION
- TTM_UPDATE
- TTM_UPDATEIPTTEXT
- TTM_WINDOWFROMPOINT
- 工具提示控件通告消息 547
 - NM_CUSTOMDRAW (工具提示)
 - TTN_GETDISPINFO
 - TTN_POP
 - TTN_SHOW
- 工具提示控件数据结构 550
 - NMTTCUSTOMDRAW
 - NMTTDISPINFO
 - TOOLINFO
 - TTHITTESTINFO
- 轨迹条控件参考 562
 - 轨迹条控件消息 562
 - TBM_CLEARSEL
 - TBM_CLEARTEXT
- TBM_GETBUDDY
- TBM_GETCHANNELRECT
- TBM_GETLINESIZE
- TBM_GETNUMTICS
- TBM_GETPAGE SIZE
- TBM_GETPOS
- TBM_GETPTICS
- TBM_GETRANGEMAX
- TBM_GETRANGEMIN
- TBM_GETSELEND
- TBM_GETSELSTART
- TBM_GETTHUMBLENGTH
- TBM_GETTHUMBRECT
- TBM_GETTIC
- TBM_GETTICPOS
- TBM_GETTOOLTIPS
- TBM_GETUNICODEFORMAT
- TBM_SETBUDDY
- TBM_SETLINESIZE
- TBM_SETPAGE SIZE
- TBM_SETPOS
- TBM_SETRANGE
- TBM_SETRANGEMAX
- TBM_SETRANGEMIN
- TBM_SETSEL
- TBM_SETSELEND
- TBM_SETSELSTART
- TBM_SETTHUMBLENGTH
- TBM SETTIC
- TBM SETTIPSIDE
- TBM SETTOOLTIPS
- 轨迹条控件通告消息 579
 - NM_CUSTOMDRAW (轨迹条)
 - NM_RELEASEDCAPTURE (轨迹条)
- 增减数控件参考 585
 - 增减数控件函数 585

CreateUpDownControl

增减数控件消息 586

UDM_GETACCEL

UDM_GETBASE

UDM_GETBUDDY

UDM_GETPOS

UDM_GETRANGE

UDM_GETRANGE32

UDM_GETUNICODEFORMAT

UDM_SETACCEL

UDM_SETBASE

UDM_SETBUDDY

UDM_SETPOS

UDM_SETRANGE

UDM_SETRANGE32

UDM_SETUNICODEFORMAT

增减数控件通告消息 593

NM_RELEASEDCAPTURE (增减数
控件)

UDN_DELTAPOS

增减数控件数据结构 594

NMUPDOWN

UDACCEL

索引B 编程元素列表

第1卷: Windows 基本服务

A

AbnormalTermination	539
AddAtom	244
AddUsersToEncryptedFile	469
AllocateUserPhysicalPages	188
AreFileApisANSI	341
AssignProcessToJobObject	53
AttachThreadInput	54

B

Beep	551
BindIoCompletionCallback	55
BY_HANDLE_FILE_INFORMATION	433

C

CallMsgFilter	298
CallNextHookEx	299
CallWndProc	300
CallWndRetProc	301
Cancello	342
CBT_CREATEWND	323
CBTACTIVATESTRUCT	323
CBTProc	302
ChangeClipboardChain	256
CHARSETINFO	581
CloseClipboard	257
CloseHandle	287
CommandLineToArgvW	56
ConvertThreadToFiber	57
COPYDATASTRUCT	242

CopyFile	343
CopyFileEx	344
CopyMemory	189
CopyProgressRoutine	345
CountClipboardFormats	257
CreateDirectory	346
CreateDirectoryEx	347
CreateFiber	57
CreateFile	348
CreateHardLink	470
CreateIoCompletionPort	355
CreateJobObject	58
CreateProcess	59
CreateProcessAsUser	66
CreateProcessWithLogonW	72
CreateRemoteThread	
CreateThread	79
CWPRETSTRUCT	324
CWPSTRUCT	324

D

DEBUGHOOKINFO	325
DebugProc	305
DecryptFile	471
DefineDosDevice	357
DeleteAtom	245
DeleteFiber	81
DeleteFile	358
DeleteVolumeMountPoint	472
DisableThreadLibraryCalls	156

DISKQUOTA_USER_INFORMATION	526
DllMain	157
DuplicateHandle	288

E

EFS_CERTIFICATE_BLOB	527
EFS_HASH_BLOB	527
EmptyClipboard	258
EncryptFile	472
ENCRIPTION_CERTIFICATE	528
ENCRIPTION_CERTIFICATE_	
HASH	528
ENCRIPTION_CERTIFICATE-HASH_	
LIST	529
ENCRIPTION_CERTIFICATE_	
LIST	529
EncryptionDisable	473
EnumClipboardFormats	258
EVENTMSG	325
EXCEPTION_POINTERS	545
EXCEPTION_RECORD	546
ExitProcess	82
ExitThread	83

F

FatalAppExit	552
FiberProc	84
FILE_NOTIFY_INFORMATION	435
FileEncryptionStatus	474
FileIOCompletionRoutine	359
FillMemory	190
FindAtom	246
FindClose	360
FindCloseChangeNotification	361
FINDEX_INFO_LEVELS	441
FINDEX_SEARCH_OPS	442
FindFirstChangeNotification	361

FindFirstFile	363
FindFirstFileEx	364
FindFirstVolume	475
FindFirstVolumeMountPoint	476
FindNextChangeNotification	367
FindNextFile	368
FindNextVolume	477
FindNextVolumeMountPoint	478
FindVolumeClose	478
FindVolumeMountPointClose	479
FlashWinlow	552
FlashWindowEx	553
FLASHWINFO	562
FlushFileBuffers	368
FONTSIGNATURE	582
ForegroundIdleProc	306
FormatMessage	554
FreeEncryptionCertificateHashList	480
FreeEnvironmentStrings	84
FreeLibrary	159
FreeLibraryAndExitThread	160
FreeUserPhysicalPages	190

G

GET_FILEEX_INFO_LEVELS	442
GetAtomName	247
GetBinaryType	369
GetClipboardData	260
GetClipboardFormatName	260
GetClipboardOwner	261
GetClipboardSequenceNumber	262
GetClipboardViewer	262
GetCommandLine	85
GetCompressedFileSize	480
GetCurrentDirectory	370
GetCurrentFiber	148
GetCurrentProcess	86

GetCurrentProcessId	86
GetCurrentThread	87
GetCurrentThreadId	88
GetDiskFreeSpace	371
GetDiskFreeSpaceEx	372
GetDriveType	373
GetEnvironmentStrings	88
GetEnvironmentVariable	89
GetExceptionCode	540
GetExceptionInformation	541
GetExitCodeProcess	90
GetExitCodeThread	90
GetFiberData	149
GetFileAttributes	374
GetFileAttributesEx	376
GetFileInformationByHandle	377
GetFileSize	378
GetFileSizeEx	378
GetFileType	379
GetFullPathName	380
GetGuiResources	91
GetHandleInformation	292
GetLastError	557
GetLogicalDrives	381
GetLogicalDriveStrings	381
GetLongPathName	382
GetModuleFileName	161
GetModuleHandle	162
GetMsgProc	307
GetOpenClipboardWindow	263
GetPriorityClass	92
GetPriorityClipboardFormat	263
GetProcAddress	163
GetProcessAffinityMask	93
GetProcessHeap	191
GetProcessHeaps	192
GetProcessIoCounters	94

GetProcessPriorityBoost	95
GetProcessShutdownParameters	95
GetProcessTimes	96
GetProcessVersion	97
GetProcessWorkingSetSize	98
GetQueuedCompletionStatus	383
GetShortPathName	385
GetStartupInfo	99
GetTempFileName	386
GetTempPath	387
GetTextCharset	571
GetTextCharsetInfo	572
GetThreadPriority	99
GetThreadPriorityBoost	100
GetThreadTimes	101
GetVolumeInformation	481
GetVolumeNameForVolumeMountPoint	484
GetVolumePathName	484
GetWriteWatch	193
GlobalAddAtom	248
GlobalDeleteAtom	249
GlobalFindAtom	249
GlobalGetAtomName	250
GlobalMemoryStatus	194

H

HeapAlloc	195
HeapCompact	196
HeapCreate	197
HeapDestroy	199
HeapFree	200
HeapLock	201
HeapReAlloc	202
HeapSize	204
HeapUnlock	205
HeapValidate	206
HeapWalk	207

I

IDiskQuotaControl	490	GetSidLength	518
AddUserName	491	Invalidate	518
AddUserSid	492	SetQuotaLimit	519
CreateEnumUsers	493	SetQuotaThreshold	520
CreateUserBatch	494	IDiskQuotaUserBatch	520
DeleteUser	495	Add	521
FindUserName	496	Remove	522
FindUserSid	497	RemoveAll	522
GetDefaultQuotaLimit	498	FlushToDisk	523
GetDefaultQuotaLimitText	499	IEnumDiskQuotaUsers	523
GetDefaultQuotaThreshold	499	Clone	524
GetDefaultQuotaThresholdText	500	Next	525
GetQuotaLogFlags	501	Reset	525
GetQuotaState	502	Skip	526
GiveUserNameResolutionPriority	502	InitAtomTable	251
Initialize	503	Int32x32To64	388
InvalidateSidNameCache	504	Int64ShlMod32	389
SetDefaultQuotaLimit	505	Int64ShraMod32	389
SetDefaultQuotaThreshold	505	Int64ShriMod32	390
SetQuotaLogFlags	506	IO_COUNTERS	133
SetQuotaState	507	IsBadCodePtr	208
ShutdownNameResolution	508	IsBadReadPtr	209
IDiskQuotaEvents	508	IsBadStringPtr	210
OnUserNameChanged	509	IsBadWritePtr	211
IDiskQuotaUser	509	IsClipboardFormatAvailable	264
GetAccountStatus	510	IsDBCSLeadByte	573
GetID	511	IsDBCSLeadByteEx	574
GetName	511	IsReparseTagHighLatency	530
GetQuotaInformation	512	IsReparseTagMicrosoft	530
GetQuotaLimit	513	IsReparseTagNameSurrogate	531
GetQuotaLimitText	514	IsTextUnicode	575
GetQuotaThreshold	514		
GetQuotaThresholdText	515		
GetQuotaUsed	516		
GetQuotaUsedText	516		
GetSid	517		

J

JOB_OBJECT_ASSOCIATE_COMPLETION_PORT	133
JOB_OBJECT_BASIC_ACCOUNTING_INFORMATION	136

JOB_OBJECT_BASIC_AND_IO_ACCOUNTING_INFORMATION	137
JOB_OBJECT_BASIC_LIMIT_INFORMATION	137
JOB_OBJECT_BASIC_PROCESS_ID_LIST	140
JOB_OBJECT_BASIC_UI_RESTRICTIONS	141
JOB_OBJECT_END_OF_JOB_TIME_INFORMATION	141
JOB_OBJECT_EXTENDED_LIMIT_INFORMATION	142
JOB_OBJECT_SECURITY_LIMIT_INFORMATION	143
JournalPlaybackProc	308
JournalRecordProc	310

K

KB_DLL_HOOK_STRUCT	326
KeyboardProc	311

L

LARGE_INTEGER	436
LoadLibrary	164
LoadLibraryEx	166
LOCALE_SIGNATURE	582
LockFile	391
LockFileEx	392
LowLevelKeyboardProc	312
LowLevelMouseProc	313

M

MAKEINTATOM	252
MapUserPhysicalPages	212
MapUserPhysicalPagesScatter	213
MEMORY_BASIC_INFORMATION	232
MEMORYSTATUS	234

MessageBeep	558
MessageProc	315
METAFILE_PICT	268
MOUSEHOOKSTRUCT	327
MOUSEHOOKSTRUCTEX	328
MouseProc	316
MoveFile	393
MoveFileEx	394
MoveFileWithProgress	397
MoveMemory	214
MSLLHOOKSTRUCT	328
MulDiv	398
MultiByteToWideChar	576

O

OFSTRUCT	436
OpenClipboard	265
OpenJobObject	102
OpenProcess	103
OpenThread	105

P

PostQueuedCompletionStatus	399
PROCESS_HEAP_ENTRY	235
PROCESS_INFORMATION	145

Q

QueryDosDevice	400
QueryInformationJobObject	106
QueryRecoveryAgentsOnEncryptedFile	485
QueryUsersOnEncryptedFile	486
QueueUserWorkItem	107

R

RaiseException	542
ReadDirectoryChangesW	401

ReadFile	404
ReadFileEx	407
ReadFileScatter	409
RegisterClipboardFormat	265
RemoveDirectory	411
RemoveUsersFromEncryptedFile	487
ReplaceFile	411
ResetWriteWatch	215
ResumeThread	108

S

SearchPath	413
SetClipboardData	266
SetClipboardViewer	267
SetCurrentDirectory	414
SetEndOfFile	415
SetEnvironmentVariable	109
SetErrorMode	559
SetFileApisToANSI	416
SetFileApisToOEM	417
SetFileAttributes	418
SetFilePointer	420
SetFilePointerEx	422
SetHandleInformation	293
SetInformationJobObject	110
SetLastError	560
SetLastErrorEx	561
SetPriorityClass	111
SetProcessAffinityMask	112
SetProcessPriorityBoost	113
SetProcessShutdownParameters	113
SetProcessWorkingSetSize	114
SetThreadAffinityMask	116
SetThreadIdealProcessor	117
SetThreadPriority	117
SetThreadPriorityBoost	119
SetUnhandledExceptionFilter	543
SetUserFileEncryptionKey	487
SetVolumeLabel	423
SetVolumeMountPoint	488
SetWindowsHookEx	317
ShellProc	319
Sleep	120
SleepEx	120
STARTUPINFO	145
SuspendThread	122
SwitchToFiber	122
SwitchToThread	123
SysMsgProc	321

T

TerminateJobObject	124
TerminateProcess	125
TerminateThread	126
TEXT	583
ThreadProc	127
TlsAlloc	127
TlsFree	128
TlsGetValue	129
TlsSetValue	130
TranslateCharsetInfo	578

U

UInt32x32To64	424
ULARGE_INTEGER	437
UnhandledExceptionFilter	545
UnhookWindowsHookEx	322
UnlockFile	425
UnlockFileEx	426
UserHandleGrantAccess	131

V

VirtualAlloc	215
VirtualAllocEx	218

VirtualFree	221
VirtualFreeEx	222
VirtualLock	224
VirtualProtect	225
VirtualProtectEx	227
VirtualQuery	229
VirtualQueryEx	230
VirtualUnlock	231

W

WaitForInputIdle	132
WideCharToMultiByte	579
WIN32_FILE_ATTRIBUTE_DATA	438
WIN32_FIND_DATA	439
WM_ASKCBFORMATNAME	269
WM_CANCELJOURNAL	329
WM_CHANGECHAIN	269
WM_CLEAR	270
WM_COPY	271

WM_COPYDATA	242
WM_CUT	271
WM_DESTROYCLIPBOARD	272
WM_DRAWCLIPBOARD	273
WM_HSCROLLCLIPBOARD	273
WM_PAINTCLIPBOARD	274
WM_PASTE	275
WM_QUEUESYNC	330
WM_RENDERALLFORMATS	276
WM_RENDERFORMAT	276
WM_SIZECLIPBOARD	277
WM_VSCROLLCLIPBOARD	278
WriteFile	427
WriteFileEx	429
WriteFileGather	432

Z

ZeroMemory	232
------------------	-----



第2卷: Windows用户接口

A

ACCEL	329
ActivateKeyboardLayout	340
AppendMenu	178

B

BlockInput	342
BM_CLICK	40
BM_GETCHECK	41
BM_GETIMAGE	42
BM_GETSTATE	43
BM_SETCHECK	43
BM_SETIMAGE	44
BM_SETSTATE	45
BM_SETSTYLE	46
BH_CLICKED	46
BH_DBLCLK	47
BH_DOUBLECLICKED	47
BH_KILLFOCUS	48
BH_SETFOCUS	49
BroadcastSystemMessage	445

C

CallWindowProc	495
CB_ADDSTRING	61
CB_DELETESTRING	62
CB_DIR	62
CB_FINDSTRING	63
CB_FINDSTRINGEXACT	64
CB_GETCOUNT	65
CB_GETCURSEL	66
CB_GETDROPPEDCONTROLRECT	66
CB_GETDROPPEDSTATE	67
CB_GETDROPPEDWIDTH	67
CB_GETEDITSEL	68
CB_GETEXTENDEDUI	69

CB_GETHORIZONTALEXTENT	69
CB_GETITEMDATA	70
CB_GETITEMHEIGHT	71
CB_GETLBTEXT	71
CB_GETLBTEXTLEN	72
CB_GETLOCALE	73
CB_GETTOPINDEX	74
CB_INITSTORAGE	74
CB_INSERTSTRING	75
CB_LIMITTEXT	76
CB_RESETCONTENT	77
CB_SELECTSTRING	77
CB_SETCURSEL	78
CB_SETDROPPEDWIDTH	79
CB_SETEXTITSEL	79
CB_SETEXTENDEDUI	80
CB_SETHORIZONTALEXTENT	81
CB_SETITEMDATA	82
CB_SETITEMHEIGHT	82
CB_SETLOCALE	83
CB_SETTOPINDEX	84
CB_SHOWDROPDOWN	84
CBN_CLOSEUP	85
CBN_DBLCLK	86
CBN_DROPDOWN	86
CBN_EDITCHANGE	87
CBN_EDITUPDATE	87
CBN_ERRSPACE	88
CBN_KILLFOCUS	89
CBN_SELCHANGE	89
CBN_SELEND CANCEL	90
CBN_SELENDOK	90
CBN_SETFOCUS	91
CharLower	234
CharLowerBuff	234
CharNext	235

CharNextExA	236
CharPrev	237
CharPrevExA	237
CharToOem	238
CharToOemBuff	239
CharUpper	240
CharUpperBuff	240
CheckDlgButton	38
CheckMenuItem	180
CheckMenuRadioItem	181
CheckRadioButton	39
ClipCursor	144
COMBOBOXINFO	55
COMPAREITEMSTRUCT	56
CompareString	241
CopyAcceleratorTable	325
CopyCursor	145
CopyIcon	157
CreateAcceleratorTable	326
CreateCaret	138
CreateCursor	146
CreateDialog	390
CreateDialogIndirect	391
CreateDialogIndirectParam	393
CreateDialogParam	394
CreateIcon	158
CreateIconFromResource	159
CreateIconFromResourceEx	160
CreateIconIndirect	162
CreateMDIWindow	474
CreateMenu	182
CreatePopupMenu	183
CURSORINFO	156

D

DefDlgProc	396
DefFrameProc	475

DefMDIChildProc	477
DefWindowProc	497
DeleteMenu	184
DestroyAcceleratorTable	326
DestroyCaret	139
DestroyCursor	147
DestroyIcon	162
DestroyMenu	184
DialogBox	397
DialogBoxIndirect	398
DialogBoxIndirectParam	399
DialogBoxParam	401
DialogProc	402
DispatchMessage	447
DlgDirListComboBox	52
DlgDirSelectComboBoxEx	54
DLGITEMTEMPLATE	422
DLGITEMTEMPLATEEX	423
DLGTEMPLATE	425
DLGTEMPLATEEX	427
DM_GETDEFID	431
DM_REPOSITION	432
DM_SETDEFID	432
DragDetect	271
DrawIcon	163
DrawIconEx	164
DRAWITEMSTRUCT	57
DrawMenuBar	185
DuplicateIcon	165

E

EnableMenuItem	186
EnableScrollBar	97
EnableWindow	342
EndDialog	403
EndMenu	187
EnumProps	498

EnumPropsEx	499
ExtractAssociatedIcon	166
ExtractIcon	167
ExtractIconEx	168

F

FoldString	244
------------------	-----

G

GET_APPCOMMAND_LPARAM	319
GET_DEVICE_LPARAM	319
GET_KEYSTATE_LPARAM	320
GET_KEYSTATE_WPARAM	320
GET_NCHITTEST_WPARAM	321
GET_WHEEL_DELTA_WPARAM	321
GET_XBUTTONDOWN_WPARAM	321
GetActiveWindow	343
GetAsyncKeyState	344
GetCapture	271
GetCaretBlinkTime	139
GetCaretPos	140
GetClipCursor	147
GetComboBoxInfo	55
GetCursor	148
GetCursorInfo	148
GetCursorPos	149
GetDialogBaseUnits	404
GetDlgCtrlID	405
GetDlgItem	406
GetDlgItemInt	406
GetDlgItemText	407
GetDoubleClickTime	272
GetFocus	345
GetIconInfo	169
GetInputState	448
GetKeyboardLayout	346
GetKeyboardLayoutList	346

GetKeyboardLayoutName	347
GetKeyboardState	348
GetKeyNameText	349
GetKeyState	350
GetLastInputInfo	351
GetMenu	187
GetMenuBarInfo	188
GetMenuCheckMarkDimensions	189
GetMenuDefaultItem	189
GetMenuInfo	190
GetMenuItemCount	191
GetMenuItemID	191
GetMenuItemInfo	192
GetMenuItemRect	193
GetMenuState	193
GetMenuString	195
GetMessage	448
GetMessageExtraInfo	450
GetMessagePos	450
GetMessageTime	451
GetMouseMovePointsEx	272
GetNextDlgGroupItem	408
GetNextDlgTabItem	409
GetProp	500
GetQueueStatus	452
GetScrollBarInfo	98
GetScrollInfo	99
GetScrollPos	100
GetScrollRange	101
GetStringTypeA	246
GetStringTypeEx	249
GetStringTypeW	252
GetSubMenu	196
GetSystemMenu	196

H

HARDWAREINPUT	372
---------------------	-----

HideCaret.....141
HiliteMenuItem197

I

ICONINFO173
ICONMETRICS174
INPUT371
InSendMessage453
InSendMessageEx453
InsertMenu198
InsertMenuItem200
IsCharAlpha.....255
IsCharAlphaNumeric256
IsCharLower256
IsCharUpper257
IsDialogMessage.....410
IsDlgButtonChecked.....40
IsMenu201
Istrcat258
Istrcmp259
Istrcmpi260
Istrcpy261
Istrcypn262
Istrlen263
IsWindowEnabled352

K

keybd_event.....352
KEYBDINPUT372
KillTimer489

L

LASTINPUTINFO373
LoadAccelerators327
LoadCursor149
LoadCursorFromFile151
LoadIcon170

LoadKeyboardLayout353
LoadMenu202
LoadMenuIndirect202
LoadString257
LookupIconIdFromDirectory171
LookupIconIdFromDirectoryEx172

M

MapDialogRect411
MapVirtualKey355
MapVirtualKeyEx356
MDICREATESTRUCT478
MDINEXTMENU215
MEASUREITEMSTRUCT59
MENUBARINFO215
MENUEX_TEMPLATE_HEADER216
MENUEX_TEMPLATE_ITEM217
MENUGETOBJECTINFO218
MENUINFO219
MenuItemFromPoint203
MENUITEMINFO220
MENUITEMTEMPLATE223
MENUITEMTEMPLATEHEADER224
MessageBox412
MessageBoxEx415
MessageBoxIndirect418
ModifyMenu204
mouse_event274
MOUSEINPUT374
MOUSEMOVEPOINT280
MSG468
MSGBOXPARAMS429

O

OemKeyScan357
OemToChar263
OemToCharBuff264

P

PeekMessage	454
PostMessage	456
PostQuitMessage	457
PostThreadMessage	457
PropEnumProc	500
PropEnumProcEx	501

Q

QueryPerformanceCounter	490
QueryPerformanceFrequency	491

R

RegisterHotKey	358
RegisterWindowMessage	459
ReleaseCapture	276
RemoveMenu	206
RemoveProp	502
ReplyMessage	459

S

SBM_ENABLE_ARROWS	113
SBM_GETPOS	114
SBM_GETRANGE	115
SBM_GETSCROLLINFO	115
SBM_SETPOS	116
SBM_SETRANGE	117
SBM_SETRANGEREDRAW	118
SBM_SETSCROLLINFO	118
SCROLLBARINFO	111
ScrollDC	103
SCROLLINFO	112
ScrollWindow	104
ScrollWindowEx	105
SendAsyncProc	460
SendDlgItemMessage	419
SendInput	359

SendMessage	461
SendMessageCallback	462
SendMessageTimeout	463
SendNotifyMessage	464
SetActiveWindow	360
SetCapture	276
SetCaretBlinkTime	141
SetCaretPos	142
SetCursor	152
SetCursorPos	153
SetDlgItemInt	420
SetDlgItemText	421
SetDoubleClickTime	277
SetFocus	361
SetKeyboardState	362
SetMenu	207
SetMenuDefaultItem	207
SetMenuInfo	208
SetMenuItemBitmaps	209
SetMenuItemInfo	210
SetMessageExtraInfo	465
SetProp	503
SetScrollInfo	107
SetScrollPos	108
SetScrollRange	109
SetSystemCursor	154
SetTimer	491
ShowCaret	143
ShowCursor	155
ShowScrollBar	110
STM_GETICON	125
STM_GETIMAGE	126
STM_SETICON	127
STM_SETIMAGE	127
STN_CLICKED	128
STN_DBLCLK	128
STN_DISABLE	129

STN_ENABLE129
SwapMouseButton278

T

TimerProc492
ToAscii362
ToAsciiEx364
ToUnicode365
ToUnicodeEx366
TPMPARAMS255
TrackMouseEvent278
TRACKMOUSEEVENT280
TrackPopupMenu211
TrackPopupMenuEx213
TranslateAccelerator328
TranslateMDISysAccel478
TranslateMessage466

U

UnloadKeyboardLayout367
UnregisterHotKey368

V

VkKeyScan369
VkKeyScanEx370

W

WaitMessage467
WindowProc497
WM_ACTIVATE376
WM_APP468
WM_APPCOMMAND282
WM_CAPTURECHANGED284
WM_CHANGEUISTATE330
WM_CHAR377
WM_COMMAND225
WM_COMPAREITEM91

WM_CONTEXTMENU226
WM_CTLCOLORBTN49
WM_CTLCOLORDLG433
WM_CTLCOLORSCROLLBAR119
WM_CTLCOLORSTATIC130
WM_DEADCHAR378
WM_DRAWITEM92
WM_ENTERIDLE434
WM_ENTERMENULOOP227
WM_ERASEBKGD175
WM_EXITMENULOOP228
WM_GETDLGCODE435
WM_GETFONT36
WM_GETHOTKEY379
WM_HOTKEY380
WM_HSCROLL120
WM_ICONERASEBKGD176
WM_INITDIALOG436
WM_INITMENU331
WM_INITMENUPOPUP332
WM_KEYDOWN381
WM_KEYUP382
WM_KILLFOCUS383
WM_LBUTTONDOWNCLK284
WM_LBUTTONDOWN286
WM_LBUTTONUP287
WM_MBUTTONDOWNCLK288
WM_MBUTTONDOWN289
WM_MBUTTONUP290
WM_MDIActivate480
WM_MDICASCADE481
WM_MDICREATE481
WM_MDIDESTROY482
WM_MDIGETACTIVE483
WM_MDIICONARRANGE484
WM_MDIIMAXIMIZE484
WM_MDINEXT485

WM_MDIREFRESHMENU	486	WM_NCXBUTTONDBLCLK	307
WM_MDIESTORE	486	WM_NCXBUTTONDOWN	309
WM_MDISETMENU	487	WM_NCXBUTTONUP	310
WM_MDI TILE	488	WM_NEXTDLGCTL	437
WM_MEASUREITEM	93	WM_NEXTMENU	231
WM_MENUCHAR	332	WM_PAINTICON	176
WM_MENUCOMMAND	228	WM_QUERYUISTATE	334
WM_MENUDRAG	229	WM_RBUTTONDBLCLK	331
WM_MENUGETOBJECT	230	WM_RBUTTONDOWN	312
WM_MENURBUTTONUP	230	WM_RBUTTONUP	313
WM_MENUSELECT	333	WM_SETCURSOR	156
WM_MOUSEACTIVATE	291	WM_SETFOCUS	384
WM_MOUSEHOVER	292	WM_SETFONT	36
WM_MOUSELEAVE	293	WM_SETHOTKEY	385
WM_MOUSEMOVE	293	WM_SYSCHAR	335
WM_MOUSEWHEEL	294	WM_SYSCOMMAND	336
WM_NCHITTEST	296	WM_SYSDEADCHAR	386
WM_NCLBUTTONDBLCLK	298	WM_SYSKEYDOWN	387
WM_NCLBUTTONDOWN	298	WM_SYSKEYUP	388
WM_NCLBUTTONUP	299	WM_TIMER	493
WM_NCMBUTTONDBLCLK	300	WM_UNINITMENUPOPUP	232
WM_NCMBBUTTONDOWN	301	WM_UPDATEUISTATE	338
WM_NCMBUTTONUP	302	WM_USER	469
WM_NCMOUSEHOVER	303	WM_VSCROLL	121
WM_NCMOUSELEAVE	303	WM_XBUTTONDBLCLK	314
WM_NCMOUSEMOVE	304	WM_XBUTTONDOWN	316
WM_NCRBUTTONDBLCLK	305	WM_XBUTTONUP	317
WM_NCRBUTTONDOWN	306	wsprintf	265
WM_NCRBUTTONUP	307	wvsprintf	267



第3卷: Windows图形设备接口

A

AbortPath	427
AlphaBlend	48
AngleArc	271
AnimatePalette	146
Arc	272
ArcTo	273

B

BeginPaint	372
BeginPath	428
BitBlt	50
BITMAP	84
BITMAPCOREHEADER	85
BITMAPCOREINFO	86
BITMAPFILEHEADER	87
BITMAPINFO	88
BITMAPINFOHEADER	89
BITMAPV4HEADER	93
BITMAPV5HEADER	96
BLENDFUNCTION	101

C

CancelDC	241
ChangeDisplaySettings	214
ChangeDisplaySettingsEx	217
Chord	258
ClientToScreen	184
CloseEnhMetaFile	289
CloseFigure	429
COLORADJUSTMENT	103
COLORREF	162
CombineRgn	465
CombineTransform	185
CopyEnhMetaFile	290

CopyRect	452
CreateBitmap	51
CreateBitmapIndirect	53
CreateBrushIndirect	114
CreateCompatibleBitmap	54
CreateCompatibleDC	219
CreateDC	220
CreateDIBitmap	55
CreateDIBPatternBrushPt	115
CreateDIBSection	56
CreateEllipticRgn	467
CreateEllipticRgnIndirect	467
CreateEnhMetaFile	291
CreateHalftonePalette	147
CreateHatchBrush	116
CreateIC	222
CreatePalette	148
CreatePatternBrush	117
CreatePen	442
CreatePenIndirect	443
CreatePolyRgn	468
CreatePolyPolygonRgn	469
CreateRectRgn	470
CreateRectRgnIndirect	470
CreatRoundRectRgn	471
CreateSolidBrush	118

D

DeleteDC	223
DeleteEnhMetaFile	293
DeleteObject	
DIBSECTION	104
DISPLAY_DEVICE	250
DPTOLP	185
DrawAnimatedRects	373

DrawCaption	374
DrawEdge	374
DrawEscape	224
DrawFocusRect	376
DrawFrameControl	377
DrawState	379
DrawStateProc	381

E

Ellipse	259
EMR	307
EMRALPHABLEND	309
EMRANGLEARC	310
EMRARCTO	311
EMRARC	311
EMRCHORD	311
EMRPIE	311
EMRBITBLT	311
EMRCOLORCORRECTPALETTE	313
EMRCOLORMATCHTOTARGET	313
EMRCREATEBRUSHINDIRECT	314
EMRCREATECOLORSPACE	315
EMRCREATECOLORSPACEW	315
EMRCREATEDIBPATTERNBRUSHPT	316
EMRCREATEMONOBRUSH	317
EMRCREATEPALETTE	318
EMRCREATEPEN	318
EMRELLIPSE	319
EMRECTANGLE	319
EMEOF	319
EMREXCLUDECLIPRECT	320
EMRINTERSECTCLIPRECT	320
EMREXTCREATEFONTINDIRECTW	320
EMREXTCREATEPEN	321
EMREXTFLOODFILL	322
EMREXTSELECTCLIPRGN	322
EMREXTTEXTOUTA	323

EMREXTTEXTOUTW	323
EMRFILLPATH	323
EMRSTROKEANDFILLPATH	323
EMRSTROKEPATH	323
EMRFILLRGN	324
EMRFORMAT	324
EMRFRAMERGN	325
EMRGDCOMMENT	326
EMRGLSBOUNDEDRECORD	326
EMRGLSRECORD	327
EMRGRADIENTFILL	327
EMRINVERTRGN	328
EMRPAINTRGN	328
EMRLINETO	329
EMRMOVETOEX	329
EMRMASKBLT	329
EMRMODIFYWORLDTRANSFORM	331
EMROFFSETCLIPRGN	332
EMRPIXELFORMAT	332
EMRPLGBLT	332
EMRPOLYDRAW	334
EMRPOLYDRAW16	335
EMRPOLYLINE	335
EMRPOLYBEZIER	335
EMRPOLYGON	335
EMRPOLYBEZIERTO	335
EMRPOLYLINETO	335
EMRPOLYLINE16	336
EMRPOLYBEZIER16	336
EMRPOLYGON16	336
EMRPOLYBEZIERTO16	336
EMRPOLYLINETO16	336
EMRPOLYPOLYLINE	337
EMRPOLYPOLYGON	337
EMRPOLYPOLYLINE16	337
EMRPOLYPOLYGON16	337
EMRPOLYTEXTOUTA	338

EMRPOLYTEXTOUTW	338
EMRRESIZEPALETTE	339
EMRSTOREDC	339
EMRROUNDRECT	340
EMRSCALEVIEWPORTEXT	340
EMRSCALEWINDOWEXT	340
EMRSELECTOBJECT	341
EMRDELETEOBJECT	341
EMRSELECTPALETTE	342
EMRSETARCDIRECTION	342
EMRSETBKCOLOR	343
EMRSETTEXTCOLOR	343
EMRSETCOLORADJUSTMENT	343
EMRSETCOLORSPACE	341
EMRSELECTCOLORSPACE	341
EMRDELETECOLORSPACE	341
EMRSETDIBITSTODEVICE	344
EMRSETICMPROFILE	345
EMRSETMAPPERFLAGS	345
EMRSETMITERLIMIT	346
EMRSETPALETTEENTRIES	346
EMRSETPIXELV	347
EMRSETVIEWPORTEXT	348
EMRSETWINDOWEXT	348
EMRSETVIEWPORTORGE	348
EMRSETWINDOWORGE	348
EMRSETBRUSHORGE	348
EMRSETWORLDTRANSFORM	349
EMRSTRETCHBLT	349
EMRSTRETCHDIBITS	350
EMRTEXT	352
EMRTRANSPARENTBLT	352
EndPaint	382
EndPath	430
无参数的增强型图元文件记录	354
有一个参数的增强型图元文件记录	354
EnhMetaFileProc	293

ENHMETAHEADER	355
ENHMETARECORD	356
EnumDisplayDevices	225
EnumDisplaySettings	226
EnumDisplaySettingsEx	227
EnumEnhMetaFile	294
EnumObjects	229
EnumObjectsProc	230
EqualRect	453
EqualRgn	472
ExcludeClipRect	128
ExcludeUpdateRgn	382
ExtCreatePen	444
ExtCreateRegion	472
ExtFloodFill	58
EXTLOGPEN	447
ExtSelectClipRgn	129

F

FillPath	430
FillRect	260
FillRgn	473
FlattenPath	431
FrameRect	261
FrameRgn	474

G

GdiComment	295
GdiFlush	383
GdiGetBatchLimit	384
GdiSetBatchLimit	385
GetArcDirection	274
GetBitmapDimensionEx	59
GetBkColor	385
GetBkMode	386
GetBoundsRect	386
GetBrushOrgEx	119

GetBValue	164
GetClipBox	130
GetClipRgn	131
GetColorAdjustment	149
GetCurrentObject	230
GetCurrentPositionEx	186
GetDC	231
GetDCBrushColor	232
GetDCEx	233
GetDCOrgEx	234
GetDCPenColor	235
GetDeviceCaps	236
GetDIBColorTable	60
GetDIBits	61
GetEnhMetaFile	297
GetEnhMetaFileBits	298
GetEnhMetaFileDescription	299
GetEnhMetaFileHeader	300
GetEnhMetaFilePaletteEntries	301
GetGraphicsMode	187
GetGValue	164
GetMapMode	187
GetMetaRgn	132
GetMiterLimit	432
GetNearestColor	149
GetNearestPaletteIndex	150
GetObject	240
GetObjectType	241
GetPaletteEntries	151
GetPath	432
GetPixel	63
GetPolyFillMode	475
GetRandomRgn	132
GetRegionData	475
GetRgnBox	476
GetROP2	387
GetRValue	165

GetStockObject	242
GetStretchBltMode	64
GetSysColorBrush	120
GetSystemPaletteEntries	152
GetSystemPaletteUse	152
GetUpdateRect	388
GetUpdateRgn	389
GetViewportExtEx	188
GetViewportOrgEx	189
GetWindowDC	390
GetWindowExtEx	190
GetWindowOrgEx	190
GetWindowRgn	391
GetWinMetaFileBits	302
GetWorldTransform	191
GRADIENT_RECT	105
GRADIENT_TRIANGLE	106
GradientFill	64
GrayString	392

H

HANDLETABLE	357
HTUI_ColorAdjustment	153

I

InflateRect	454
IntersectClipRect	133
IntersectRect	454
InvalidateRect	393
InvalidateRgn	394
InvertRect	262
InvertRgn	477
IsRectEmpty	455

L

LineDDA	275
LineDDAProc	276

LineTo	277
LoadBitmap	66
LockWindowUpdate	395
LOGBRUSH	122
LOGBRUSH32	124
LOGPALETTE	162
LOGPEN	449
LPtoDP	191

M

MAKEPOINTS	461
MAKEROP4	109
MapWindowPoints	192
MaskBlt	67
ModifyWorldTransform	193
MoveToEx	277

O

OffsetClipRgn	134
OffsetRect	455
OffsetRgn	477
OffsetViewportOrgEx	194
OffsetWindowOrgEx	195
OutputProc	396

P

PaintDesktop	397
PaintRgn	478
PAINTSTRUCT	407
PALETTEENTRY	163
PALETTEINDEX	165
PALETTEINDEX	165
PALETTEINDEX	166
PatBlt	120
PathToRegion	434
Pie	262
PlayEnhMetaFile	303

PlayEnhMetaFileRecord	304
PlgBlt	69
POINT	460
POINTL	357
POINTS	460
POINTSTOPOINT	462
PolyBezier	278
PolyBezierTo	279
PolyDraw	280
Polygon	263
Polyline	281
PolylineTo	282
PolyPolygon	264
PolyPolyline	283
PtInRect	456
PtInRegion	479
PtVisible	135

R

RealizePalette	154
RECT	461
Rectangle	265
RECTL	358
RectInRegion	479
RectVisible	136
RedrawWindow	397
ReleaseDC	244
ResetDC	245
ResizePalette	155
RestoreDC	245
RGB	166
RGBQUAD	107
RGBTRIPLE	107
RGNDATA	482
RGNDATAHEADER	482
RoundRect	266

S

SaveDC	246
ScaleViewportExtEx	196
ScaleWindowExtEx	197
ScreenToClient	198
SelectClipPath	136
SelectClipRgn	137
SelectObject	247
SelectPalette	156
SetArcDirection	283
SetBitmapDimensionEx	71
SetBkColor	399
SetBkMode	400
SetBoundsRect	401
SetBrushOrgEx	121
SetColorAdjustment	157
SetDCBrushColor	248
SetDCPenColor	249
SetDIBColorTable	71
SetDIBits	73
SetDIBitsToDevice	74
SetEnhMetaFileBits	305
SetGraphicsMode	198
SetMapMode	200
SetMetaRgn	138
SetMiterLimit	435
SetPaletteEntries	158
SetPixel	76
SetPixelV	77
SetPolyFillMode	480
SetRect	457
SetRectEmpty	458
SetRectRgn	481
SetROP2	402
SetStretchBltMode	78
SetSystemPaletteUse	159
SetViewportExtEx	201

SetViewportOrgEx	202
SetWindowExtEx	203
SetWindowOrgEx	204
SetWindowRgn	403
SetWinMetaFileBits	306
SetWorldTransform	205
SIZE	108
StretchBlt	79
StretchDIBits	81
StrokeAndFillPath	435
StrokePath	436
SubtractRect	458

T

TransparentBlt	83
TRIVERTEX	108

U

UnionRect	459
UnrealizeObject	160
UpdateColors	161
UpdateWindow	404

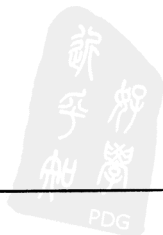
V

ValidateRect	405
ValidateRgn	406
VIDEOPARAMETERS	251

W

WidenPath	437
WindowFromDC	406
WM_DEVMODECHANGE	254
WM_DISPLAYCHANGE	408
WM_NCPAINT	408
WM_PAINT	409
WM_PALETTECHANGED	167
WM_PALETTEISCHANGING	168

WM_PRINT.....	411	WM_SYSCOLORCHANGE	169
WM_PRINTCLIENT	412	X	
WM_QUERYNEWPALETTE	169	XFORM	206
WM_SETREDRAW	412		
WM_SYNCPAINT	413		



第4卷: Windows通用控件

A

ACM_OPEN	97
ACM_PLAY	98
ACM_STOP	99
ACN_START	104
ACN_STOP	104
AddPropSheetPageProc	346
Animate_Close	99
Animate_Create	100
Animate_Open	101
Animate_OpenEx	101
Animate_Play	102
Animate_Seek	103
Animate_Stop	104

C

CBEM_DELETEITEM	113
CBEM_GETCOMBOCONTROL	114
CBEM_GETEDITCONTROL	114
CBEM_GETEXTENDEDSTYLE	115
CBEM_GETIMAGELIST	115
CBEM_GETITEM	115
CBEM_GETUNICODEFORMAT	116
CBEM_HASEDITCHANGED	116
CBEM_INSERTITEM	117
CBEM_SETEXTENDEDSTYLE	118
CBEM_SETIMAGELIST	118
CBEM_SETITEM	119
CBEM_SETUNICODEFORMAT	119
CBEN_BEGINEDIT	120
CBEN_DELETEITEM	120
CBEN_DRAGBEGIN	121
CBEN_ENDEDIT	121
CBEN_GETDISPINFO	122
CBEN_INSERTITEM	122
CCM_GETUNICODEFORMAT	65

CCM_GETVERSION	67
CCM_SETUNICODEFORMAT	67
CCM_SETVERSION	68
COLORSCHEME	80
COMBOBOXEXITEM	123
CreatePropertySheetPage	346
CreateStatusWindow	445
CreateUpDownControl	585

D

DateTime_GetMonthCal	162
DateTime_GetMonthCalColor	162
DateTime_GetMonthCalFont	163
DateTime_GetRange	164
DateTime_GetSystemtime	164
DateTime_SetFormat	165
DateTime_SetMonthCalColor	166
DateTime_SetMonthCalFont	166
DateTime_SetRange	167
DateTime_SetSystemtime	168
DestroyPropertySheetPage	347
DL_BEGINDRAG	180
DL_CANCELDRAG	181
DL_DRAGGING	182
DL_DROPPED	182
DRAGLISTINFO	183
DrawInsert	179
DrawStatusText	446
DTM_GETMCCOLOR	155
DTM_GETMCFONT	156
DTM_GETMONTHCAL	156
DTM_GETRANGE	157
DTM_GETSYSTEMTIME	158
DTM_SETFORMAT	158
DTM_SETMCCOLOR	159
DTM_SETMCFONT	160

DTM_SETRANGE	160
DTM_SETSYSTEMTIME	161
DTN_CLOSEUP	168
DTN_DATETIMECHANGE	169
DTN_DROPDOWN	170
DTN_FORMAT	170
DTN_FORMATQUERY	171
DTN_USERSTRING	172
DTN_WMKEYDOWN	172

E

ExtensionPropSheetPageProc	347
----------------------------------	-----

F

FIRST_IPADDRESS	258
FlatSB_EnableScrollBar	187
FlatSB_GetScrollInfo	188
FlatSB_GetScrollPos	189
FlatSB_GetScrollProp	190
FlatSB_GetScrollRange	191
FlatSB_SetScrollInfo	192
FlatSB_SetScrollPos	193
FlatSB_SetScrollProp	194
FlatSB_SetScrollRange	196
FlatSB_ShowScrollBar	197
FORWARD_WM_NOTIFY	71
FOURTH_IPADDRESS	258

G

GetEffectiveClientRect	62
GetMUILanguage	62

H

HANDLE_WM_NOTIFY	71
HDHITTESTINFO	239
HDITEM	240
HDLAYOUT	243

HDM_CLEARFILTER	205
HDM_CREATEDRAGIMAGE	206
HDM_DELETEITEM	206
HDM_EDITFILTER	207
HDM_GETBITMAPMARGIN	207
HDM_GETIMAGELIST	208
HDM_GETITEM	208
HDM_GETITEMCOUNT	209
HDM_GETITEMRECT	209
HDM_GETORDERARRAY	210
HDM_GETUNICODEFORMAT	211
HDM_HITTEST	211
HDM_INSERTITEM	212
HDM_LAYOUT	212
HDM_ORDERTOINDEX	212
HDM_SETBITMAPMARGIN	213
HDM_SETFILTERCHANGETIMEOUT	214
HDM_SETHOTDIVIDER	214
HDM_SETIMAGELIST	215
HDM_SETITEM	216
HDM_SETORDERARRAY	216
HDM_SETUNICODEFORMAT	217
HDN_BEGINDRAG	231
HDN_BEGINTRACK	231
HDN_DIVIDERDBLCLICK	232
HDN_ENDDRAG	232
HDN_ENDTRACK	233
HDN_FILTERBTNCLICK	233
HDN_FILTERCHANGE	234
HDN_GETDISPINFO	235
HDN_ITEMCHANGED	235
HDN_ITEMCHANGING	236
HDN_ITEMCLICK	236
HDN_ITEMDBLCLICK	237
HDN_TRACK	237
HDTEXTFILTER数据结构	243
Header_ClearFilter	217

Header_CreateDragImage	218
Header_DeleteItem	219
Header_EditFilter	219
Header_GetBitmapMargin	220
Header_GetImageList	220
Header_GetItem	221
Header_GetItemCount	222
Header_GetItemRect	222
Header_GetOrderArray	223
Header_GetUnicodeFormat	224
Header_InsertItem	224
Header_Layout	225
Header_OrderToIndex	226
Header_SetBitmapMargin	226
Header_SetFilterChangeTimeout	227
Header_SetHotDivider	227
Header_SetImageList	228
Header_SetItem	229
Header_SetOrderArray	230
Header_SetUnicodeFormat	230
HKM_GETHOTKEY	251
HKM_SETHOTKEY	251
HKM_SETRULES	252

I

INDEXTOSTATEIMAGE MASK	72
InitCommonControls	63
InitCommonControlsEx	63
INITCOMMONCONTROLSEX	80
InitializeFlatSB	186
InitMUILanguage	64
IPM_CLEARADDRESS	254
IPM_GETADDRESS	254
IPM_ISBLANK	255
IPM_SETADDRESS	255
IPM_SETFOCUS	256
IPM_SETRANGE	256

IPM_FIELDCHANGED	257
------------------------	-----

L

LBItemFromPt	179
--------------------	-----

M

MakeDragList	180
MAKEIPADDRESS	259
MAKEIPRANGE	259
MCHITTESTINFO	303
MCM_GETCOLOR	269
MCM_GETCURSEL	269
MCM_GETFIRSTDAYOFWEEK	270
MCM_GETMAXSEL COUNT	271
MCM_GETMAXTODAYWIDTH	271
MCM_GETMINREQRECT	271
MCM_GETMONTHDELTA	272
MCM_GETMONTHRANGE	273
MCM_GETRANGE	274
MCM_GETSELRANGE	274
MCM_GETTODAY	275
MCM_GETUNICODEFORMAT	276
MCM_HITTEST	276
MCM_SETCOLOR	278
MCM_SETCURSEL	279
MCM_SETDAYSTATE	279
MCM_SETFIRSTDAYOFWEEK	280
MCM_SETMAXSEL COUNT	281
MCM_SETMONTHDELTA	281
MCM_SETRANGE	282
MCM_SETSELRANGE	283
MCM_SETTODAY	283
MCM_SETUNICODEFORMAT	284
MCN_GETDAYSTATE	300
MCN_SELCHANGE	301
MCN_SELECT	302
MenuHelp	447

MonthCal_GetColor	285
MonthCal_GetCurSel	285
MonthCal_GetFirstDayOfWeek	286
MonthCal_GetMaxSelCount	287
MonthCal_GetMaxTodayWidth	287
MonthCal_GetMinReqRect	288
MonthCal_GetMonthDelta	289
MonthCal_GetMonthRange	289
MonthCal_GetRange	290
MonthCal_GetSelRange	291
MonthCal_GetToday	291
MonthCal_GetUnicodeFormat	292
MonthCal_HitTest	293
MfOnthCal_SetColor	293
MonthCal_SetCurSel	294
MonthCal_SetDayState	295
MonthCal_SetFirstDayOfWeek	296
MonthCal_SetMaxSelCount	296
MonthCal_SetMonthDelta	297
MonthCal_SetRange	298
MonthCal_SetSelRange	298
MonthCal_SetToday	299
MonthCal_SetUnicodeFormat	300
MONTHDAYSTATE	305

N

NM_CHAR	73
NM_CLICK	73
NM_CLICK (状态条)	458
NM_CLICK (选项卡)	507
NM_CUSTOMDRAW	90
NM_CUSTOMDRAW (头标)	237
NM_CUSTOMDRAW (rebar)	425
NM_CUSTOMDRAW (工具提示)	547
NM_CUSTOMDRAW (轨迹条)	579
NM_DBLCLK	74
NM_DBLCLK (状态条)	458

NM_HOVER	74
NM_KEYDOWN	74
NM_KILLFOCUS	75
NM_KILLFOCUS (date time)	173
NM_NCHITTEST	75
NM_NCHITTEST (rebar)	426
NM_OUTOFMEMORY	76
NM_RCLICK	76
NM_RCLICK (头标)	238
NM_RCLICK (状态条)	459
NM_RCLICK (选项卡)	507
NM_RDBLCLK	77
NM_RDBLCLK (状态条)	459
NM_RELEASEDCAPTURE	77
NM_RELEASEDCAPTURE (头标)	239
NM_RELEASEDCAPTURE (monthcal)	302
NM_RELEASEDCAPTURE (pager)	323
NM_RELEASEDCAPTURE (rebar)	427
NM_RELEASEDCAPTURE (选项卡)	507
NM_RELEASEDCAPTURE (轨迹条)	581
NM_RELEASEDCAPTURE (增减数 控件)	593
NM_RETURN	78
NM_SETCURSOR	78
NM_SETCURSOR (扩展组合框)	123
NM_SETFOCUS	79
NM_SETFOCUS (date time)	123
NM_TOOLTIPS_CREATED	79
NMCBEDRAGBEGIN	126
NMCBEENDEDIT	125
NMCHAR	81
NMCOMBOBOXEX	126
NMCUSTOMDRAW	92
NMDATETIMECHANGE	174
NMDATETIMEFORMAT	174
NMDATETIMEFORMATQUERY	175
NMDATETIMESTRING	176

NMDATETIMEWMKEYDOWN	177
NMDAYSTATE	304
NMHDDISPINFO	244
NMHDFILTERBTNCLICK数据结构	244
NMHDR	82
NMHEADER	245
NMIPADDRESS	261
NMKEY	82
NM_MOUSE	83
NMOBJECTNOTIFY	83
NMPGALCSIZE	324
NMPGSCROLL	325
NMRBAUTOSIZE	433
NMREBAR	433
NMREBARCHEVRON	434
NMREBARCHILDSize	435
NMSELCHANGE	305
NMTCKEYDOWN	510
NMTOOLTIPS_CREATED	84
NMTTCUSTOMDRAW	550
NMTTDISPINFO	550
NMUPDOWN	594

P

Pager_ForwardMouse	315
Pager_GetBkColor	316
Pager_GetBorder	317
Pager_GetButtonSize	317
Pager_GetButtonState	318
Pager_GetDropTarget	318
Pager_GetPos	319
Pager_RecalcSize	319
Pager_SetBkColor	320
Pager_SetBorder	320
Pager_SetButtonSize	321
Pager_SetChild	322
Pager_SetPos	322

PBM_DELTAPOS	330
PBM_GETPOS	331
PBM_GETRANGE	331
PBM_SETBARCOLOR	332
PBM_SETBKCOLOR	332
PBM_SETPOS	333
PBM_SETRANGE	333
PBM_SETRANGE32	334
PBM_SETSTEP	334
PBM_STEPIT	335
PBRANGE	335
PGM_FORWARDMOUSE	308
PGM_GETBKCOLOR	309
PGM_GETBORDER	309
PGM_GETBUTTONSIZE	310
PGM_GETBUTTONSTATE	310
PGM_GETDROPTARGET	311
PGM_GETPOS	312
PGM_RECALCSIZE	312
PGM_SETBKCOLOR	312
PGM_SETBORDER	313
PGM_SETBUTTONSIZE	314
PGM_SETCHILD	314
PGM_SETPOS	315
PGM_CALCSIZE	323
PGN_SCROLL	324
PropertySheet	348
PropSheet_AddPage	366
PropSheet_Apply	367
PropSheet_CancelToClose	367
PropSheet_Changed	368
PropSheet_GetCurrentPageHwnd	368
PropSheet_GetTabControl	369
PropSheet_HwndToIndex	369
PropSheet_IdToIndex	370
PropSheet_IndexToHwnd	371
PropSheet_IndexToId	371

PropSheet_IndexToPage	372
PropSheet_InsertPage	373
PropSheet_IsDialogMessage	374
PropSheet_PageToIndexg	374
PropSheet_PressButton	375
PropSheet_QuerySiblings	376
PropSheet_RebootSystem	376
PropSheet_RemovePage	377
PropSheet_RestartWindows	377
PropSheet_SetCurSel	378
PropSheet_SetCurSelByID	379
PropSheet_SetFinishText	379
PropSheet_SetHeaderSubTitle	380
PropSheet_SetHeaderTitle	380
PropSheet_SetTitle	381
PropSheet_SetWizButtons	382
PropSheet_UnChanged	383
PROPSHEETHEADER	392
PROPSHEETPAGE	396
PropSheetPageProc	349
PropSheetProc	350
PSHNOTIFY	399
PSM_ADDPAGE	351
PSM_APPLY	351
PSM_CANCELTOCLOSE	352
PSM_CHANGED	352
PSM_GETCURRENTPAGEHWND	353
PSM_GETTABCONTROL	353
PSM_HWNDTOINDEX	354
PSM_IDTOINDEX	354
PSM_INDEXTOHWND	355
PSM_INDEXTOID	355
PSM_INDEXTOPAGE	356
PSM_INSERTPAGE	356
PSM_ISDIALOGMESSAGE	357
PSM_PAGETOINDEX	358
PSM_PRESSBUTTON	358

PSM_QUERYSIBLINGS	359
PSM_REBOOTSYSTEM	359
PSM_REMOVEPAGE	360
PSM_RESTARTWINDOWS	361
PSM_SETCURSEL	361
PSM_SETCURSELID	362
PSM_SETFINISHTEXT	362
PSM_SETHEADERSUBTITLE	363
PSM_SETHEADERTITLE	363
PSM_SETTITLE	364
PSM_SETWIZBUTTONS	364
PSM_UNCHANGED	366
PSN_APPLY	383
PSN_GETOBJECT	384
PSN_HELP	385
PSN_KILLACTIVE	385
PSN_QUERYCANCEL	386
PSN_QUERYINITIALFOCUS	387
PSN_RESET	387
PSN_SETACTIVE	388
PSN_TRANSLATEACCELERATOR	389
PSN_WIZBACK	390
PSN_WIZFINISH	390
PSN_WIZNEXT	391

R

RB_BEGINDRAG	405
RB_DELETEBAND	406
RB_DRAGMOVE	406
RB_ENDDRAG	407
RB_GETBANDBORDERS	407
RB_GETBANDCOUNT	408
RB_GETBANDINFO	408
RB_GETBARHEIGHT	409
RB_GETBARINFO	409
RB_GETBKCOLOR	410
RB_GETCOLORSCHEME	410

RB_GETDROPTARGET	411	REBARBANDINFO	436
RB_GETPALETTE	411	REBARINFO	439
RB_GETRECT	412		
RB_GETROWCOUNT	412	S	
RB_GETROWHEIGHT	413	SB_GETBORDERS	448
RB_GETTEXTCOLOR	413	SB_GETICON	448
RB_GETTOOLTIPS	414	SB_GETPARTS	449
RB_GETUNICODEFORMAT	414	SB_GETRECT	449
RB_HITTEST	415	SB_GETTEXT	450
RB_IDTOINDEX	415	SB_GETTEXTLENGTH	451
RB_INSERTBAND	416	SB_GETTIPTTEXT	451
RB_MAXIMIZEBAND	416	SB_GETUNICODEFORMAT	452
RB_MINIMIZEBAND	417	SB_ISSIMPLE	452
RB_MOVEBAND	417	SB_SETBKCOLOR	453
RB_PUSHCHEVRON	418	SB_SETICON	453
RB_SETBANDINFO	419	SB_SETMINHEIGHT	454
RB_SETBARINFO	419	SB_SETPARTS	454
RB_SETBKCOLOR	420	SB_SETTEXT	455
RB_SETCOLORSCHEME	421	SB_SETTIPTTEXT	456
RB_SETPALETTE	421	SB_SETUNICODEFORMAT	456
RB_SETPARENT	422	SB_SIMPLE	457
RB_SETTEXTCOLOR	422	SBN_SIMPLEMODECHANGE	459
RB_SETTOOLTIPS	423	SECOND_IPADDRESS	260
RB_SETUNICODEFORMAT	423	ShowHideMenuCtl	65
RB_SHOWBAND	424		
RB_SIZE TO RECT	425	T	
RBHITTESTINFO	436	TabCtrl_AdjustRect	491
RBN_AUTOSIZE	427	TabCtrl_DeleteAllItems	491
RBN_BEGINDRAG	428	TabCtrl_Deleteltem	492
RBN_CHEVRONPUSHED	428	TabCtrl_DeselectAll	492
RBN_CHILDSIZE	429	TabCtrl_GetCurFocus	493
RBN_DELETEDBAND	429	TabCtrl_GetCurSel	493
RBN_DELETINGBAND	430	TabCtrl_GetExtendedStyle	494
RBN_ENDDRAG	430	TabCtrl_GetImageList	494
RBN_GETOBJECT	431	TabCtrl_GetItem	495
RBN_HEIGHTCHANGE	431	TabCtrl_GetItemCount	495
RBN_LAYOUTCHANGED	432	TabCtrl_GetItemRect	496

TabCtrl_GetRowCount	496	TBM_SETBUDDY	571
TabCtrl_GetToolTips	497	TBM_SETLINESIZE	572
TabCtrl_GetUnicodeFormat	497	TBM_SETPAGESIZE	572
TabCtrl_HighlightItem	498	TBM_SETPOS	573
TabCtrl_HitTest	499	TBM_SETRANGE	574
TabCtrl_InsertItem	499	TBM_SETRANGEMAX	574
TabCtrl_RemoveImage	500	TBM_SETRANGEMIN	575
TabCtrl_SetCurFocus	500	TBM_SETSEL	575
TabCtrl_SetCurSel	501	TBM_SETSELEND	576
TabCtrl_SetExtendedStyle	501	TBM_SETSELSTART	576
TabCtrl_SetImageList	502	TBM_SETTHUMBLENGTH	577
TabCtrl_SetItem	503	TBM_SETTIC	577
TabCtrl_SetItemExtra	503	TBM_SETTIPSID	578
TabCtrl_SetItemSize	504	TBM_SETTOOLTIPS	579
TabCtrl_SetMinTabWidth	504	TCHITTESTINFO	511
TabCtrl_SetPadding	505	TCITEM	512
TabCtrl_SetToolTips	505	TCITEMHEADER	513
TabCtrl_SetUnicodeFormat	506	TCM_ADJUSTRECT	476
TBM_CLEARSEL	562	TCM_DELETEALLITEMS	477
TBM_CLEARTICS	563	TCM_DELETEITEM	477
TBM_GETBUDDY	563	TCM_DESELECTALL	478
TBM_GETCHANNELRECT	564	TCM_GETCURFOCUS	478
TBM_GETLINESIZE	564	TCM_GETCURSEL	479
TBM_GETNUMTICS	565	TCM_GETEXTENDEDSTYLE	479
TBM_GETPAGESIZE	565	TCM_GETIMAGELIST	479
TBM_GETPOS	566	TCM_GETITEM	480
TBM_GETPTICS	566	TCM_GETITEMCOUNT	480
TBM_GETRANGEMAX	567	TCM_GETITEMRECT	481
TBM_GETRANGEMIN	567	TCM_GETROWCOUNT	481
TBM_GETSELEND	567	TCM_GETTOOLTIPS	482
TBM_GETSELSTART	568	TCM_GETUNICODEFORMAT	482
TBM_GETTHUMBLENGTH	568	TCM_HIGHLIGHTITEM	483
TBM_GETTHUMBRECT	569	TCM_HITTEST	483
TBM_GETTIC	569	TCM_INSERTITEM	484
TBM_GETTICPOS	570	TCM_REMOVEIMAGE	484
TBM_GETTOOLTIPS	570	TCM_SETCURFOCUS	485
TBM_GETUNICODEFORMAT	571	TCM_SETCURSEL	485

TCM_SETEXTENDEDSTYLE486
 TCM_SETIMAGELIST487
 TCM_SETITEM487
 TCM_SETITEMEXTRA488
 TCM_SETITEMSIZE488
 TCM_SETMINTABWIDTH489
 TCM_SETPADDING489
 TCM_SETTOOLTIPS490
 TCM_SETUNICODEFORMAT490
 TCN_FOCUSCHANGE508
 TCN_GETOBJECT508
 TCN_KEYDOWN509
 TCN_SELCHANGE509
 TCN_SELCHANGING510
 THIRD_IPADDRESS260
 TOOLINFO552
 TTHITTESTINFO553
 TTM_ACTIVATE530
 TTM_ADDTOOL530
 TTM_ADJUSTRECT531
 TTM_DELTOOL532
 TTM_ENUMTOOLS532
 TTM_GETBUBBLESIZE533
 TTM_GETCURRENTTOOL533
 TTM_GETDELAYTIME534
 TTM_GETMARGIN534
 TTM_GETMAXTIPWIDTH535
 TTM_GETTEXT536
 TTM_GETTIPBKCOLOR536
 TTM_GETTOOLCOUNT537
 TTM_GETTOOLINFO537
 TTM_HITTEST538
 TTM_NEWTOOLRECT538
 TTM_POP539
 TTM_RELAYEVENT539
 TTM_SETDELAYTIME540
 TTM_SETMARGIN540

TTM_SETMAXTIPWIDTH541
 TTM SETTIPBKCOLOR542
 TTM SETTIPTEXTCOLOR542
 TTM_SETTITLE543
 TTM_SETTOOLINFO544
 TTM_TRACKACTIVATE544
 TTM_TRACKPOSITION545
 TTM_UPDATE545
 TTM_UPDATETIPTTEXT546
 TTM_WINDOWFIROMPOINT546
 TTN_GETDISPINFO548
 TTN_POP549
 TTN_SHOW549

U

UDACCEL595
 UDM_GETACCEL586
 UDM_GETBASE587
 UDM_GETBUDDY587
 UDM_GETPOS588
 UDM_GETRANGE588
 UDM_GETRANGE32589
 UDM_GETUNICODEFORMAT589
 UDM_SETACCEL590
 UDM_SETBASE590
 UDM_SETBUDDY591
 UDM_SETPOS591
 UDM_SETRANGE592
 UDM_SETRANGE32592
 UDM_SETUNICODEFORMAT593
 UDN_DELTAPOS594
 UninitializeFlatSB198

W

WM_NOTIFY69
 WM_NOTIFYFORMAT69

第5卷: Windows Shell

A

ABM_ACTIVATE	543
ABM_GETAUTOHIDEBAR	543
ABM_GETSTATE	544
ABM_GETTASKBARPOS	544
ABM_NEW	544
ABM_QUERYPOS	545
ABM_REMOVE	545
ABM_SETAUTOHIDEBAR	546
ABM_SETPOS	546
ABM_WINDOWPOSCHANGED	547
ABN_FULLSCREENAPP	547
ABN_POSCHANGED	548
ABN_STATECHANGE	548
ABN_WINDOWARRANGE	549
AssocCreate	505
ASSOCDATA	421
ASSOCF	421
ASSOCKEY	422
AssocQueryKey	505
AssocQueryString	506
AssocQueryStringByKey	507
ASSOCSTR	422

B

BrowseCallbackProc	362
--------------------------	-----

C

ChrCmpl	432
ColorAdjustLuma	532
ColorHLSToRGB	533
ColorRGBToHLS	534
CPL_DBLCLK	549
CPL_EXIT	550
CPL_GETCOUNT	550
CPL_INIT	551

CPL_INQUIRE	551
CPL_NEWINQUIRE	552
CPL_STARTWPARMS	553
CPL_STOP	553
CPIApplet	309

D

DefScreenSaverProc	309
DllGetVersion	310
DllGETVERSIONPROC	311
DllInstall	535
DoEnvironmentSubst	311
DragAcceptFiles	313
DragFinish	313
DragQueryFile	314
DragQueryPoint	314

F

FindEnvironmentString	315
FindExecutable	316
FM_GETDRIVEINFO	554
FM_GETFILESEL	555
FM_GETFILESELLFN	555
FM_GETFOCUS	556
FM_GETSELCOUNT	556
FM_GETSELCOUNTLFN	556
FM_REFRESH_WINDOWS	557
FM_RELOAD_EXTENSIONS	557
FMEVENT_HELPMENUITEM	558
FMEVENT_HELPSTRING	558
FMEVENT_INITMENU	559
FMEVENT_LOAD	559
FMEVENT_SELCHANGE	560
FMEVENT_TOOLBARLOAD	560
FMEVENT_UNLOAD	561
FMEVENT_USER_REFRESH	561

FMExtensionProc	363
FOLDERFLAGS	423
FOLDERVIEWMODE	424

G

GetMenuContextHelpI	317
GetWindowContextHelpI	317

H

HashData	536
----------------	-----

I

IAList	
Expand	105
IAList2	
GetOptions	106
SetOptions	107
IActiveDesktop	
AddDesktopItem方法	108
AddDesktopItemWithUI方法	109
AddUri方法	110
ApplyChanges	111
GenerateDesktopItemHtml	112
GetDesktopItem	112
GetDesktopItemByID	113
GetDesktopItemBySource	113
GetPattern	114
GetDesktopItemCount	115
GetDesktopItemOptions	115
GetWallpaper	115
GetWallpaperOptions	116
ModifyDesktopItem	116
RemoveDesktopItem	117
SetDesktopItemOptions	118
SetPattern	118
SetWallpaper	119
SetWallpaperOptions	119

IAAsyncOperation

EndOperation	120
GetAsyncMode	121
InOperation	122
SetAsyncMode	122
StartOperation	123

IAutoComplete

Enable	125
Init	126

IAutoComplete2

GetOptions	127
SetOptions	128

IColumnProvider

GetColumnInfo	130
GetItemData	131
Initialize	131

ICommDlgBrowser

IncludeObject	132
OnDefaultCommand	133
OnStateChange	134

ICommDlgBrowser2

GetDefaultMenuText	135
GetViewFlags	136
Notify	136

IContextMenu

GetCommandString	137
InvokeCommand	138
QueryContextMenu	139

IContextMenu2

HandleMenuMsg	142
---------------------	-----

IContextMenu3

HandleMenuMsg2	143
----------------------	-----

ICopyHook

CopyCallback	145
--------------------	-----

ICurrentWorkingDirectory

GetDirectory	146
SetDirectory	147

IDeskBand		Reset	174
GetBandInfo	148	Skip	174
IDockingWindow		IEnumIDList	
CloseDW	149	Clone	175
ResizeBorderDW	150	Next	176
ShowDW	151	Reset	177
IDockingWindowFrame		Skip	177
AddToolBar	152	IExtractIcon	
FindToolBar	153	Extract	178
RemoveToolBar	153	GetIconLocation	179
IDockingWindowSite		IExtractImage	
GetBorderDW	160	Extract	181
RequestBorderSpaceDW	161	GetLocation	182
SetBorderSpaceDW	162	IExtractImage2	
IDragSourceHelper		GetDateStamp	184
InitializeFromBitmap	155	IFileViewer	
InitializeFromWindow	155	PrintTo	185
IDropTargetHelper		Show	185
DragEnter	157	ShowInitialize	186
DragLeave	158	IFileViewerSite	
DragOver	158	GetPinnedWindow	187
Drop	159	SetPinnedWindow	188
Show	159	IInputObject	
IEmptyVolumeCache		HasFocusIO	189
Deactivate	163	TranslateAcceleratorIO	189
GetSpaceUsed	163	UIActivateIO	190
Initialize	164	IInputObjectSite	
Purge	166	OnFocusChangeIS	191
ShowProperties	167	InetIsOffline	318
IEmptyVolumeCache2		INewShortcutHook	
InitializeEx	168	GetExtension	192
IEmptyVolumeCacheCallback		GetFolder	192
PurgeProgress	171	GetName	193
ScanProgress	172	GetReferent	193
IEnumExtraSearch		SetFolder	194
Clone	173	SetReferent	194
Next	173	INotifyReplica	

- YouAreAReplica195
- IntlStrEqN432
- IntlStrEqNI433
- IntlStrEqWorker434
- IObjMar
 - Append196
 - Remove197
- IPersistFileSystemFolder
 - GetFolderTargetInfo200
 - InitializeEx200
- IPersistFolder
 - Initialize198
- IPersistFolder2
 - GetCurFolder199
- IProgressDialog
 - HasUserCancelled203
 - SetAnimation203
 - SetCancelMsg204
 - SetLine204
 - SetProgress205
 - SetProgress64206
 - SetTitle206
 - StartProgressDialog207
 - StopProgressDialog208
 - Timer208
- IQueryAssociations
 - GetData210
 - GetEnum211
 - GetKey211
 - GetString212
 - Init213
- IQueryInfo
 - GetInfoFlags214
 - GetInfoTip215
- IReconcilableObject
 - GetProgressFeedbackMaxEstimate216
 - Reconcile217
- IReconcileInitiator
 - SetAbortCallback220
 - SetProgressFeedback221
- IRemoteComputer
 - Initialize222
- IResolveShellLink
 - ResolveShellLink223
- IRunnableTask
 - IsRunning225
 - Kill226
 - Resume226
 - Run226
 - Suspend227
- IShellBrowser
 - BrowseObject228
 - EnableModelessSB229
 - GetControlWindow230
 - GetViewStateStream231
 - InsertMenusSB232
 - OnViewWindowActive233
 - QueryActiveShellView233
 - RemoveMenusSB234
 - SendControlMsg235
 - SetMenuSB235
 - SetStatusTextSB236
 - SetToolbarItems237
 - TranslateAcceleratorSB237
- IShellChangeNotify
 - OnChange239
- IShellDetails
 - ColumnClick241
 - GetlDetailsOf242
- IShellExecuteHook
 - Execute244
- IShellExtInit
 - Initialize245
- IShellFolder

BindToObject	247	SetArguments	278
BindToStorage	248	SetDescription	279
CompareIDs	249	SetHotkey	279
CreateViewObject	250	SetIconLocation	280
EnumObjects	250	SetIDList	280
GetAttributesOf	251	SetPath	281
GetDisplayNameOf	253	SetRelativePath	281
GetUIObjectOf	254	SetShowCmd	282
ParseDisplayName	255	SetWorkingDirectory	283
SetNameOf	255		
IShellFolder2		IShellLinkDataList	
EnumSearches	260	AddDataBlock	284
GetDefaultColumn	260	CopyDataBlock	285
GetDefaultColumnState	261	GetFlags	285
GetDefaultSearchGUID	262	RemoveDataBlock	286
GetDetailsEx	262	SetFlags	287
GetDetailsOf	263		
MapNameToSCID	264	IShellPropSheetExt	
IShellIcon		AddPages	288
GetIconOf	265	ReplacePage	289
IShellIconOverlay		IShellView	
GetOverlayIconIndex	267	AddPropertySheetPages	290
GetOverlayIndex	267	CreateViewWindow	291
IShellIconOverlayIdentifier		DestroyViewWindow	292
GetOverlayInfo	269	EnableModeless	292
GetPriority	270	EnableModelessSV	293
IsMemberOf	270	GetCurrentInfo	293
IShellLink		GetItemObject	294
GetArguments	272	Refresh	294
GetDescription	272	SaveViewState	295
GetHotkey	273	SelectItem	296
GetIconLocation	274	TranslateAccelerator	296
GetIDList	274	UIActivate	297
GetPath	275	IShellView2	
GetShowCmd	275	CreateViewWindow2	299
GetWorkingDirectory	276	GetView	299
Resolve	277	HandleRename	300
		SelectAndPositionItem	300
		ITaskbarList	

ActivateTab	302	PathFindSuffixArray	468
AddTab	302	PathGetArgs	468
DeleteTab	303	PathGetCharType	469
HrInit	303	PathGetDriveNumber	469
SetActiveAlt	304	PathIsContentType	470
Uniform ResourceLocator		PathIsDirectory	470
GetURL	305	PathIsDirectoryEmpty	471
InvokeCommand	306	PathIsFileSpec	471
SetURL	306	PathIsHTMLFile	472
IURL_SETURL_FLAGS	425	PathIsLFNFileSpec	472
IURL_SETURL_INVOKECOMMAND_		PathIsNetworkPath	473
FLAGS	425	PathIsPrefix	474
IURLSearchHook		PathIsRelative	474
Translate	308	PathIsRoot	475
M		PathIsSameRoot	475
MAKEDLLVERULL	428	PathIsSystemFolder	476
MIMEAssociationDialog	318	PathIsUNC	476
MLLoadLibrary	434	PathIsUNCServer	477
P		PathIsUNCServerShare	477
PathAddBackslash	458	PathIsURL	478
PathAddExtension	459	PathMakePretty	478
PathAppend	460	PathMakeSystemFolder	479
PathBuildRoot	460	PathMatchSpec	479
PathCanonicalize	461	PathParseIconLocation	480
PathCombine	461	PathQuoteSpaces	481
PathCommonPrefix	462	PathRelativePathTo	481
PathCompactPath	463	PathRemoveArgs	482
PathCompactPathEx	463	PathRemoveBackslash	483
PathCreateFromUrl	464	PathRemoveBlanks	483
PathFileExists	465	PathRemoveExtension	484
PathFindExtension	465	PathRemoveFileSpec	484
PathFindFileName	466	PathRenameExtension	485
PathFindNextComponent	466	PathSearchAndQualify	485
PathFindOnPath	467	PathSetDlgItemPath	486
		PathSkipRoot	486
		PathStripPath	487

PathStripToRoot	487
PathUndecorate	488
PathUnExpandEnvStrings	488
PathUnmakeSystemFolder	490
PathUnquoteSpaces	490

R

RegisterDialogClasses	319
REGSAM	504

S

ScreenSaverConfigureDialog	320
ScreenSaverProc	321
SetMenuContextHelpId	322
SetWindowContextHelpId	322
SHAddTocRecentDocs	323
SHAppBarMessage	324
SHAutoComplete	536
SHBindToParent	324
SHBrowseForFolder	326
SHChangeNotify	326
SHCONTF	426
SHCopyKey	508
SHCreateDirectoryEx	329
SHCreateProcessAsUser	330
SHCreateShellPalette	534
SHCreateStreamOnFile	358
SHCreateThread	538
SHDeleteEmptyKey	509
SHDeleteKey	510
SHDeleteValue	511
Shell_NotifyIcon	331
ShellAbout	332
ShellExecute	333
ShellExecuteEx	336
SHEmptyRecycleBin	337
SHEnumKeyEx	511
SHEnumValue	512
SHFileOperation	338
SHFreeNameMappings	339
SHGetDataFromIDLList	339
SHGetDesktopFolder	340
SHGetDiskFreeSpace	341
SHGetFileInfo	342
SHGetFolderLocation	344
SHGetFolderPath	346
SHGetIconOverlayIndex	347
SHGetInstanceExplorer	348
SHGetMalloc	349
SHGetNewLinkInfo	350
SHGetPathFromIDLList	351
SHGetSettings	352
SHGetSpecialFolderLocation	353
SHGetSpecialFolderPath	353
SHGetThreadRef	539
SHGetValue	513
SHGNO	427
SHInvokePrinterCommand	354
SHLoadInProc	356
SHOpenRegStream	540
SHOpenRegStream2	541
SHQueryInfoKey	514
SHQueryRecycleBin	356
SHQueryValueEx	515
SHRegCloseUSKey	516
SHRegCreateUSKey	517
SHREGDEL_FLAGS	531
SHRegDeleteEmptyUSKey	518
SHRegDeleteUSValue	518
SHRegDuplicateHKey	519
SHREGENUM_FLAGS	532
SHRegEnumUSKey	520
SHRegEnumUSValue	520
SHRegGetBoolUSValue	521

SHRegGetPath	522
SHRegGetUSValue	523
SHRegOpenUSKey	524
SHRegQueryInfoUSKey	525
SHRegQueryUSValue	526
SHRegSetPath	527
SHRegSetUSValue	528
SHRegWriteUSValue	529
SHSetThreadRef	542
SHSetValue	530
SHStrDup	435
SOANGLETENTHS	429
SoftwareUpdateMessageBox	357
SOPALETTEINDEX	430
SOPALETTERGB	430
SORGB	430
SOSETRATIO	431
StrCat	436
StrCatBuff	437
StrChr	437
StrChrI	438
StrCmp	439
StrCmpl	439
StrCmpN	440
StrCmpNI	441
StrCpy	442
StrCpyN	442
StrCSpn	443
StrCSpnI	443
StrDup	444
StrFormatByteSize	445
StrFormatByteSize64A	446
StrFormatKBSize	447
StrFromTimeInterval	447
StrIsIntlEqual	448
StrNCat	449
StrPBrk	450

StrRChr	450
StrRChrI	451
StrRetToBuf	451
StrRetToStr	452
StrRStrI	453
StrSpn	454
StrStr	454
StrStrI	455
StrToInt	455
StrToIntEx	456
StrTrim	456

T

TranslateURL	358
TRANSLATEURL_IN_FLAGS	427

U

UndeleteFile	364
UrlApplyScheme	490
URLAssociationDialog	359
URLASSOCIATIONDIALOG_IN_FLAGS	428
UrlCanonicalize	492
UrlCombine	492
UrlCompare	493
UrlCreateFromPath	494
UrlEscape	495
UrlEscapeSpaces	496
UrlGetLocation	496
UrlGetPart	497
UrlHash	498
Urlls	499
UrllsFileUrl	500
UrllsNoHistory	500
UrllsOpaque	501
UrlUnEscape	502
UrlUnEscapeInPlace	502

W

WinHelp	360	WM_HELP	563
WM_CPLL_AUNCH	562	WM_TCARD	564
WM_CPL_LAUNCHED	562	wnsprintf	457
WM_DROPFILES	562	wvnsprintf	458