(54) Title: METHOD FOR MANIPULATING A LIVE AUDIO MEDIA STREAM

(57) Abstract: A method for manipulating a live audio media stream. A live audio media stream is created and encoded from a news text that has been parsed into individual components, such as head line, news release text, stock symbols, and industry code. The components are stored for comparing to user inputted preferences. Upon a user computer receiving the live audio media stream, the media stream is decoded and if the industry codes and stock symbols match the users preferences, the user computer media player volume is turned on and the headline is displayed by highlighted text in a scrolling table. The headline text is also an active URL link to the full text of the press release on a web site. When the next release is received and there is no match of the preferences the volume of the player is returned to its initial state and the headline is displayed in normal text.
METHOD FOR MANIPULATING A LIVE AUDIO MEDIA STREAM

FIELD OF THE INVENTION

This invention relates to communication systems using the Internet and in particular to a method for manipulating a live audio media stream providing an on-line real time custom tailored news source.

BACKGROUND OF THE INVENTION

Every business day hundreds of millions of shares of stock representing thousands of publicly owned companies are traded by individual and institutional investors. These publicly owned companies represent a variety of industry sectors, e.g. transportation, utilities, energy, telecommunications, computer technology, retail, real estate, and health.

In years past many individual investors relied on full service brokerage firms, mutual fund managers, or investment advisers for making investment decisions. But that has changed with the growing popularity of the Internet, on-line trading, and the availability of on-line investment information, many investors are choosing to become informed and make their own decisions.

Since the stock market is known to be volatile, investors are well aware of the importance of knowing the latest financial and business news, and, in particular, news about their holdings or about stocks they are considering to buy. Obviously, this news is important because of its potential affect on the value of the investors holdings in these companies.

Daily news on stock market activity is available through a variety of sources, such as television, radio, newspaper, newswire, the Internet, and various electronic push technologies employing computers, pagers, and Personal Digital Assistants (PDA). These sources fall into the general categories of broadcast, print, and electronic media. News items may concern the market in general, an
industry sector, or a corporation. The news might be statistical data on an industry sector cyclical trend, or a corporate announcement about a new product, merger, or change in executive management. The news item sometimes includes the trading price of one or more related stocks.

Broadcast news reports (i.e. television and radio) are real time and occur at scheduled intervals. A viewer or listener must tune in with undivided attention in order to receive the information. Unless the investor records the broadcast, the information generally cannot be retrieved. Other problems with broadcast news are the time constraints and large target audience, which tend to result in highlight news of broad appeal. For the serious investor, real-time broadcast news does not usually provide enough detail about the stocks of interest, and therefore is not a preferred source of investment news.

Print media, such as newspapers, provide the previous day’s trading prices for thousands of stocks and in-depth business reports. However, in addition to the problem of day-old information, the investor must devote full attention to manually searching for news that is of interest. Due to limited print space, business reports are selected based on relevance to many readers and do not provide 100% coverage of business news.

The news wire probably provides the most recent business news that can be easily accessed through electronic means. News wire services are generally the primary source of national and world business news for the broadcast and print media. Therefore news is typically available through a news wire service before it is broadcast or printed. News items are stored in computer databases and can be easily searched. However, the investor must actively perform the search and retrieve the information, diverting time and attention from other tasks.
The Internet gives the investor access to many sources of investment information. Many financially oriented web sites provide useful services to an investor. Some of these enable a user to build a customized watch list of stock symbols, showing trading information such as price of the last trade (usually delayed 15-20 minutes), day's high and low, volume, and historical price data. Also provided are automatic links to news and research for each stock on the watch list. Many of the news links are obtained from the newswire services and are specific to that stock or its industry sector. Although the information is at the user's fingertips, it still requires time and attention to review the news links periodically to see if any new items have appeared in the list.

There are other services that employ push technology to deliver information directly to the investor. One service delivers customized headline news and stock data passively to a user's computer desktop and is accessed by opening an icon. Other services provide stock market quote information via a two-way pager, Personal Digital Assistant (PDA), or e-mail. With this service users can schedule quotes at a specified time for a desired list of symbols and get price alerts when the quote for a specified symbol has reached the target price.

The fundamental problem common to all of these news sources is that the investor must dedicate a certain amount of time and attention to obtain the investment news of interest. For most investors, monitoring and searching for relevant investment news is not their primary occupation. While a large number of people interested in investment news have office jobs with access to a computer and the Internet, monitoring and searching for investment news detracts from their everyday occupational responsibilities and tasks.
Thus, what is lacking in the art is a method for
manipulating a live audio media stream providing an on-line
real time custom tailored news source.

SUMMARY OF THE INVENTION

The invention is a method for manipulating a live audio
media stream such as, by way of example, a PRNewswire press
release. In this example, the press release is received by
a News Server computer is read until one whole press release
has been received. The press release story is then sent to
a PRNewswire Parser. The PRNewswire Parser parses through
the press release to break apart the data into its
individual components, i.e. headline, press release text,
stock symbols, industry codes etc. This data is put into a
press release object and sent to server software.

The server software receives the press release object
and stores each of the fields of the object into a record in
the database. The release is also added to an editor queue.
Once in the queue it is sent to the editor computer where an
editor client application adds the release to a list box.
The server also manages the narrator queue, ordering the
releases in the queue by priority code. This queue is
populated after an editor has edited and saved a press
release on one of the editor computers. Another main
function of the server is to keep track of the status of all
the press releases as they flow through the system, updating
the statuses in the database as they change.

The Editor Client is the interface between an editor
and the server. When an editor executes an operation (edit,
skip, delete, save, or cancel) on a press release, the
Editor Client makes the appropriate method call on the
server software to handle the operation. When server
software handles any operation on a press release it must
push information to the Editor Client to inform the Editor
Client when the status of a press release has changed so
that the Editor Client can update its information to the editor. The Narrator Client gets the next press release to be read from the server upon the narrator “pressing” the next button on the console. Upon receiving the release from the server a call is made to the media encoder, such as a Microsoft Windows Media Encoder, to send a command and text script string through the live media stream. The data sent in the stream for the “HEADLINE” type includes the stock symbol associated with the release, the industry code associated with the release, the headline of the release and the unique ID of the record in the database for this release. There are also buttons on the console to send commands and text strings for market update, end of day and commercials.

On the web server there are Active Server Pages that manage the user registration, user preferences and user sign-in. When a user visits a web site they are asked to register to gain access to the service. Once the consumer has registered they can create their preferences. In the preferred embodiment, the user is allowed to select up to 10 industry codes and 25 stock symbols to have on their “watch list”. These preferences are stored in the user’s record in the database and stored in “cookies” in the web browser on the user’s computer. On subsequent sign-ins the preferences are retrieved from the database and stored in the “cookies”.

On the user’s computer when the user “launches” a player the Javascript on the HTML page reads the preferences from the “cookie” and stores them in local variables; a media player is also started. When text string and command that was sent by the media encoder is received by the media player, the player calls a Javascript function that determines what type of command has been received. The commands that are currently implemented are "HEADLINE", "MRKTUPDT" (market update), "EOD" (end of day) and "PROMO" (for commercials). When the “HEADLINE” command is received,
the text is decoded to get the id, stock symbol, industry
code and headline text for the headline that is about to be
read by the narrator. The stock symbols and industry code
of the release are compared against the preferences set by
the user. If a match is found then a call is made to the
media player to make sure the volume is turned on if the
user has chosen to set it to mute. The stock symbol and
headline are also displayed highlighted in a scrolling
table. The headline text is also an active (Uniform
Resource Locator) URL link to the full text of the press
release on the web site. When the next release is received
and there is no match of the preferences the volume of the
player is returned to its previous state and the headline is
displayed in normal text. When the other commands and text
strings are received the volume is turned on if it is set to
mute and a message is displayed to the user. The user can
also adjust the volume level of the player by using the up
and down arrow buttons on the player.

A n objective of the invention is to disclose a method
for manipulating a live audio media stream.

A nother objective of the invention is to disclose a
method for providing an on-line real time custom tailored
news source.

Sti l another objective of the invention is to provide
a news source that will automatically provide an investor
with critical news on selected holdings by parsing a news
text feed into individual components and providing queue to
determine a priority order of media releases.

A nother objective of the invention is to compare media
releases to user preferences and, manipulate the volume of a
media player based upon matching of user preferences and
also highlighting the headline of said media release when
matched against the user preferences.

Sti l another objective of the invention is to provide
an active URL link to the full text of a media release.
Other objectives and advantages of this invention will become apparent from the following description taken in conjunction with the accompanying drawings wherein are set forth, by way of illustration and example, certain embodiments of this invention. The drawings constitute a part of this specification and include exemplary embodiments of the present invention and illustrate various objects and features thereof.

**BRIEF DESCRIPTION OF THE DRAWINGS**

FIG. 1 is a data flow diagram which illustrates a communication system according to the preferred embodiment of the present invention.

**DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT**

Although the invention has been described in terms of a specific embodiment, it will be readily apparent to those skilled in this art that various modifications, rearrangements and substitutions can be made without departing from the spirit of the invention. The scope of the invention is defined by the claims appended hereto.

As illustrated in FIG.1, a communication system in accordance with the preferred embodiment includes, but is not limited to, the following components: a server computer 10, a database 12, a plurality of editor computers 14, 16 & 18, a narrator computer 20, a media server 22, a user computer 24, and a Web site server 26. The server computer 10 includes communication means 28 with an external data provider 30. In addition, the server computer 10 is connected to the database 12, the editor computers 14, 16 & 18 and the narrator computer 20. The narrator computer 20 also includes a communication means 34 with the windows media server 22. The windows media server 22 includes a communication means 36 with the user computer 24. The user computer 24 includes a communication means 30 with the Web
site server 26. The Web site server 26 includes a communication means 40 with the database 20.

Within the server computer 10 are several software modules which perform processes necessary to this invention. The feed manager 42 receives a press release text transmission from an external data provider 30, reads in one complete press release story by use of the following module:

FeedManager.vbp
- frmTerminal.frm
- modFeedManager.mod
- cFeed.cls
- clsNotify.cls

frmTerminal.frm

Option Explicit
Dim filename
Dim Ret As Integer     ' Scratch integer.
Dim Temp As String     ' Scratch string.
Dim StartTime As Date  ' Stores starting time for port timer
Dim oNotifyParse As clsNotify
Dim characteramount As Integer

Private Sub Form_Load()
On Error GoTo Err_frm
Dim CommPort As String, Handshaking As String,
Settings As String
Picture1.ZOrder
On Error Resume Next
App.Title = "PRS News Feed Manager"
'8/31/99
filename = FreeFile
Open "c:\check.txt" For Append As #filename
Write #filename, "I'm in frmTerminal onLoad"
Close #filename

txtTerm.SelLength = Len(txtTerm)
txtTerm.SelText = ""
flag = False
Buffer = ""
Buffer2 = ""
Set oNotifyParse = New clsNotify
numrecs = 0
characteramount = 0
sbrStatus.Panels("Status").Text = "Status:
Disconnect "imgNotConnected.ZOrder
   frmTerminal.Move (Screen.Width - Width) / 2,
   (Screen.Height - Height) / 2
   Settings = GetSetting(App.Title, "Properties",
"Settings", ") MSComm1.Settings"

If Settings <> "" Then
   MSComm1.Settings = Settings
   If Err Then
      MsgBox Error$, 48
      Exit Sub
   End If
End If

CommPort = GetSetting(App.Title, "Properties",
"CommPort", ") MSComm1.CommPort
If CommPort <> "" Then MSComm1.CommPort = 2

Handshaking = GetSetting(App.Title, "Properties",
"Handshaking", ") MSComm1.Handshaking
   If Handshaking <> "" Then
      MSComm1.Handshaking = Handshaking
      If Err Then
         MsgBox Error$, 48
         Exit Sub
      End If
   End If
'Tim modification to make no echo
   'Echo = GetSetting(App.Title, "Properties",
"Echo", ") Echo
   Echo = False
   mnuOpen.Click
   'Tim modification to make echo false on startup
   'mnuEcho.Checked = True
   mnuEcho.Checked = False
Exit Sub

Err_Form:
   sRetCode = "Line Number: " & Erl & " Err: " &
   Err.Number & " Error: " & Err.Description
   oServerEvent.ReportEvent strcomputername,
   "FeedManager", "frmTerminal", "Form_Load", sRetCode
End Sub

Private Sub Form_Resize()
   On Error GoTo Err_Resize
   ' Resize the Term (display) control
   txtTerm.Move 0, tbrToolBar.Height,
   frmTerminal.ScaleWidth, frmTerminal.ScaleHeight -
   sbrStatus.Height-tbrToolBar.Height

Err_Resize:
Frame1.Left = ScaleWidth - Frame1.Width * 1.5
Frame2.Left = ScaleWidth - Frame1.Width * 4.5
Exit Sub
Err.Resize:
  If Err.Number = 5 Then
    Resume Next
  End If
  sRetCode = "Line Number: " & Erl & " Error: " &
  Err.Number & " Error: " & Err.Description
  oServerEvent.ReportEvent strcomputername,
  "FeedManager", "frmTerminal", "Form_Resizing", sRetCode
End Sub

Private Sub Form_Unload(Cancel As Integer)
On Error GoTo Err_FormUnload
  Dim Counter As Long
  Set oNotifyParse = Nothing
  Set oServerEvent = Nothing
  If MSComm1.PortOpen Then
    MSComm1.PortOpen = 0
  End If
  Exit Sub
Err_FormUnload:
  sRetCode = "Line Number: " & Erl & " Error: " &
  Err.Number & " Error: " & Err.Description
  oServerEvent.ReportEvent strcomputername,
  "FeedManager", "frmTerminal", "Form_UnLoad", sRetCode
End Sub

Private Sub imgConnected_Click()
  ' Call the mnuOpen_Click routine to toggle connect
  and disconnect
  Call mnuOpen_Click
End Sub

Private Sub imgNotConnected_Click()
  Call mnuOpen_Click
End Sub

Private Sub mnuAbout_Click()
  frmAbout.Show vbModal
End Sub

Private Sub mnuEcho_Click()
  mnuEcho.Checked = Not mnuEcho.Checked
  txtTerm = ""
End Sub

Private Sub mnuFileExit_Click()
Unload frmTerminal
End Sub

Private Sub mnuProperties_Click()
    frmProperties.Show vbModal
End Sub

' Toggles the state of the port (open or closed).
Private Sub mnuOpen_Click()
    On Error Resume Next
    Dim OpenFlag
    MSComm1.PortOpen = Not MSComm1.PortOpen
    If Err Then MsgBox Err$, 48
    OpenFlag = MSComm1.PortOpen
    mnuOpen.Checked = OpenFlag
    If MSComm1.PortOpen Then
        imgConnected.ZOrder
        sbrStatus.Panels("Settings").Text = "Settings:"
        " & MSComm1.Settings
        sbrStatus.Panels("Status").Text = "Status:
        Connect "
        StartTiming
        Else
        imgNotConnected.ZOrder
        sbrStatus.Panels("Settings").Text = "Settings:"
        "
        sbrStatus.Panels("Status").Text = "Status:
        Disconnect "
        StopTiming
        End If
    End Sub

' The OnComm event is used for trapping communications events and errors.
Private Static Sub MSComm1_OnComm()
    On Error GoTo Err_MSComm1
    Select Case MSComm1.CommEvent
        ' Event messages.
        Case comEvReceive
            Buffer = MSComm1.Input
            If Asc(StrConv(Buffer, vbUnicode)) = SOH
                characteramount = 0
            Then
Buffer2 = ""
Picture2.ZOrder = True
End If
If flag And Asc(StrConv(Buffer,
vbUnicode)) <> EOT Then
    If characteramount > 32000 Then
        flag = False
        Picture1.ZOrder = True
    End If
    Buffer2 = Buffer2 & StrConv(Buffer,
    vbUnicode)
    characteramount = characteramount +
    1
End If
If flag And Asc(StrConv(Buffer,
    vbUnicode)) = EOT Then
    Picture1.ZOrder = False
    Buffer2 = Buffer2 & StrConv(Buffer,
    vbUnicode)
    oNotifyParse.Parse StrConv(Buffer2,
    vbFromUnicode)
    Write #filenum, "I'm sending a
    record"
    Close #filenum
End If
If mnuEcho.Checked Then
    ShowData txtTerm, StrConv(Buffer,
    vbUnicode)
End If

End Select
Exit Sub
Err_MSComm1:
    sRetCode = "Line Number: " & Erl & " Err: " &
Err.Number & " Error: " & Err.Description
    oServerEvent.ReportEvent strcomputername,
    "FeedManager", "frmTerminal", "MSComm1_OnComm", sRetCode
    MsgBox Err.Number & " " & Err.Description
End Sub

Private Static Sub ShowData(Term As Control, Data As String)
    On Error GoTo Handler
    Const MAXTERMSIZE = 16000
    Dim TermSize As Long, i
    

' Make sure the existing text doesn't get too large.
TermSize = Len(Term.Text)
If TermSize > MAXTERMSIZE Then
  Term.Text = Mid$(Term.Text, 4097)
  TermSize = Len(Term.Text)
End If

' Point to the end of Term's data.
Term.SelStart = TermSize

' Filter/handle BACKSPACE characters.
Do
  i = InStr(Data, Chr$(8))
  If i Then
    If i = 1 Then
      Term.SelStart = TermSize - 1
      Term.SellLength = 1
      Data = Mid$(Data, i + 1)
    Else
      Data = Left$(Data, i - 2) & Mid$(Data, i + 1)
    End If
  End If
Loop While i

' Eliminate line feeds.
Do
  i = InStr(Data, Chr$(10))
  If i Then
    Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
  End If
Loop While i

' Make sure all carriage returns have a line feed.
i = 1
Do
  i = InStr(i, Data, Chr$(13))
  If i Then
    Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
    i = i + 1
  End If
Loop While i
Term.SelText = Data
Term.SelStart = Len(Term.Text)
Exit Sub

Handler:
MsgBox Error$
Resume Next
End Sub
Private Sub Timer2_Timer()
    sbrStatus.Panels("Status").Text = "Status: "
    Timer2.Enabled = False
End Sub

Private Sub tbrToolBar_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
    Case "Properties"
        Call mnuProperties_Click
    End Select
End Sub

Private Sub Timer1_Timer()
    ' Display the Connect Time
    sbrStatus.Panels("Stories").Text = "Records Processed: " & numrecs & 
    sbrStatus.Panels("ConnectTime").Text = Format(Now - StartTime, "hh:nn:ss") & "  
    sbrStatus.Panels("Characters").Text = characteramount
End Sub

    ' Call this function to start the Connect Time timer
Private Sub StartTiming()
    StartTime = Now
    Timer1.Enabled = True
End Sub

Private Sub StopTiming()
    Timer1.Enabled = False
    sbrStatus.Panels("ConnectTime").Text = ""
End Sub

----modFeedManager.mod

' Public variables
Public Const SOH = 1
Public Const EOT = 4
Public Echo As Boolean ' Echo On/Off flag.
Public CancelSend As Integer ' Flag to stop sending a text file
Public numrecs As Integer
Public flag As Boolean
Public Buffer As Variant
Public Buffer2 As Variant
Public sRetCode As String
Public oServerEvent As Object
Public strcomputername As String * 20
Declare Sub SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal
Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long
Public Declare Function GetComputerName Lib "kernel32"
Alias "GetComputerNameA" (ByVal lpBuffer As String, nSize As Long) As Long

Sub Main()
On Error GoTo Err_Main
Dim rc As Integer
10 rc = GetComputerName(strcomputername,
Len(strcomputername))
20 Set oServerEvent =
CreateObject("NewswireManager.cEventLog")
30 frmTerminal.Show
Exit Sub
Err_Main:
_msgBox "Line Number: " & Erl & " Error Number: " &
Err.Number & " " & Err.Description
sRetCode = "Line Number: " & Erl & " Err: " &
Err.Number & " Error: " & Err.Description
oServerEvent.ReportEvent strcomputername,
"FeedManager", "modFeedManager", "Sub Main", sRetCode
End Sub

------cFeed.cls
Option Explicit

Dim tBuffer As String * 1000
Dim tBufferStatus As String * 1
Public Property Get BufferStatus() As String
BufferStatus = tBufferStatus
End Property

Public Property Let BufferStatus(ByVal vNewValue As String)
tBufferStatus = vNewValue
End Property

Public Property Get Buffer() As String
    Buffer = tBuffer
End Property

Public Property Let Buffer(ByVal vNewValue As String)
tBuffer = vNewValue
End Property

------clsNotify.cls
Option Explicit
Public WithEvents x As FeedParser.cParse

Private Sub x.Notify(flag As Boolean)
On Error GoTo Err.Notify
If Not flag Then
    numrecs = numrecs + 1
End If
Set x = Nothing
Exit Sub
Err.Notify:
    Resume Next
End Sub

Public Sub Parse(strtoparse As String)
On Error GoTo Err.Parse
    Set x = New FeedParser.cParse
    x.ParseString strtoparse
    Exit Sub
Err.Parse:
End Sub

Private Sub Class_Terminate()
    Set x = Nothing
End Sub

(Copyright 1999 VISUAL DATA CORPORATION)

The module forwards the complete story to the feed parser 44 in accordance with the following module:
FeedParser.vbp
    - modparse.bas
    - cParse.cls
modParse.bas
Option Explicit

Public Const SOH = 1
Public Const STX = 2
Public Const ETX = 3
Public Const EOT = 4
Public Const LF = 10
Public Const CR = 13
Public Const CS1 = 17
Public Const DC3 = 19
Public Const US = 31
Public strcomputername As String * 20
Public oServerEvent As Object
Public oFeedParser As cParse
Public oStory As Newswire1.cANPA
Public oServerManager As Object
Public sRetCode As String
Public strTest As String
Public Declare Function GetComputerName Lib "kernel32"
    Alias "GetComputerNameA" (ByVal lpBuffer As String,
        nSize As Long) As Long

Public Sub Parse(ByRef oFeedParser As cParse,
    strtoparse As String)
    On Error GoTo Err_Parse

10      Dim readpos As Integer, endpos As Integer
20      Dim str As String, strjunk As String
30      Dim chrstr As String
40      Dim roleresult As String
50      Dim filenumber

45      filenumber = FreeFile

50      Set oStory = New Newswire1.cANPA
60      readpos = 2
70      endpos = InStr(readpos, strtoparse, Chr$(US))
80      oStory.ServiceLevelDes = CStr(Mid$(strtoparse,
90      readpos, endpos - readpos))
95      "MsgBox "Service Level Des=" &
100     oStory.ServiceLevelDes
110     readpos = endpos + 1
120     endpos = InStr(readpos, strtoparse, Chr$(LF))
130     oStory.FiveDigit = Mid(strtoparse, readpos,
140     endpos - readpos)
145     "MsgBox "FiveDigit=" & oStory.FiveDigit
150     readpos = endpos + 1
160     endpos = InStr(readpos, strtoparse, ",")
165     oStory.Priority = Mid(strtoparse, readpos, endpos
170     "MsgBox "Priority=" & oStory.Priority
180     readpos = endpos + 1
190     endpos = InStr(readpos, strtoparse, Chr$(DC3))
200     oStory.Category = Mid(strtoparse, readpos, endpos
205     "MsgBox "Category=" & oStory.Category
210     endpos = InStr(readpos, strtoparse, "bc-")
220     readpos = endpos + 3
240    endpos = InStr(readpos, strtoparse, """)
250    oStory.StoryKeyword = Mid(strtoparse, readpos, endpos - readpos)
260    oStory.StoryKeyword = Replace(oStory.StoryKeyword, "", ",
270    'MsgBox "Keyword=" & oStory.StoryKeyword
280    readpos = endpos + 1
290    endpos = InStr(readpos, strtoparse, """)
300    oStory.Version = Mid(strtoparse, readpos, endpos - readpos)
320    'MsgBox "Version=" & oStory.Version
330    readpos = endpos + 1
340    endpos = InStr(readpos, strtoparse, """)
350    oStory.Reference = Mid(strtoparse, readpos, endpos - readpos)
360    oStory.Reference = Replace(oStory.Reference, "", ",
370    'MsgBox "Reference=" & oStory.Reference
380    readpos = endpos + 1
390    endpos = InStr(readpos, strtoparse, vbCrLf)
400    oStory.TransmissionDate = Mid(strtoparse, readpos, endpos - readpos)
410    'MsgBox "TDATE=" & oStory.TransmissionDate
420    readpos = endpos + 4
430    endpos = InStr(readpos, strtoparse, ")"
440    oStory.BureauBanner = Mid(strtoparse, readpos, endpos - readpos)
450    oStory.BureauBanner = Replace(oStory.BureauBanner, "", ",
460    'MsgBox "BureauBanner=" & oStory.BureauBanner
470    endpos = InStr(endpos, strtoparse, "]STK]")
480    readpos = endpos + 5
490    endpos = InStr(readpos, strtoparse, vbCrLf)
500    oStory.StockSymbols = Mid(strtoparse, readpos, endpos - readpos)
510    'MsgBox "StockSymbols=" & oStory.StockSymbols
520    endpos = InStr(readpos, strtoparse, "]IN")
530    readpos = endpos + 4
540    endpos = InStr(readpos, strtoparse, vbCrLf)
550    oStory.IndustryCode = Mid(strtoparse, readpos, endpos - readpos)
560    'MsgBox "IndustryCode=" & oStory.IndustryCode
570    endpos = InStr(readpos, strtoparse, "]SU")
580    readpos = endpos + 4
590    endpos = InStr(readpos, strtoparse, vbCrLf)
600    oStory.SubjectCode = Mid(strtoparse, readpos, endpos - readpos)
610    'MsgBox "SubjectCode=" & oStory.SubjectCode
620    readpos = endpos + 2
endpos = InStr(readpos, strtoparse, ":")
doStory.Addressing = Mid(strtoparse, readpos, endpos - readpos)
oStory.Addressing = Replace(oStory.Addressing, ":", ":", TextCompare)
MsgBox "Addressing=" & oStory.Addressing
dpos = endpos + 5 'double space between
Addressing and Headline, skip CRLF
readpos = endpos
endpos = InStr(readpos, strtoparse, vbCrLf & vbCrLf)
oStory.Headline = Mid(strtoparse, readpos, endpos - readpos)
oStory.Headline = Replace(oStory.Headline, ":", ":", TextCompare)
*****************************************************************************9/01/99 Under
specifications Correctly.......
MsgBox "Headline=" & oStory.Headline
endpos = endpos + 4
readpos = endpos
endpos = InStr(readpos, strtoparse, "SOURCE", vbBinaryCompare)
If endpos = 0 Then
Err.Raise 1, , "Ignore Error: Source does not
exist in the story."
'Exit Sub
End If
oStory.Paragraphs = Mid(strtoparse, readpos, endpos - readpos)
oStory.Paragraphs = Replace(oStory.Paragraphs, ":", ":", TextCompare)
MsgBox "Paragraphs=" & oStory.Paragraphs
readpos = endpos + 6
endpos = InStr(readpos, strtoparse, vbCrLf)
oStory.Source = Mid(strtoparse, readpos, readpos - endpos)
oStory.Source = Replace(oStory.Source, ":", ":", TextCompare)
*****************************************************************************9/01/99 Correct
implementation of specs up to this point
readpos = endpos + 9
endpos = endpos + 9
'changed 890 from endpos + 10 to endpos + 9
'the backup copy in Feedparser folder will have
nonchanged version.
While Not IsNumeric(Mid(strtoparse, endpos, 1))
endpos = endpos + 1
Wend
readpos = endpos 'At beginning digit of
AvailabilityDate
chrstr = Mid(strtoparse, endpos, 1)
950 While chrstr <> " " And chrstr <> vbCr
960     endpos = endpos + 1
970     chrstr = Mid(strtoparse, endpos, 1)
980 Wend
990 oStory.AvailabilityDate = Mid(strtoparse, readpos, endpos - readpos) "AvailabilityDate"
1000 If chrstr = " " Then
1010     endpos = endpos + 1
1020     readpos = endpos
1030     oStory.AvailabilityStatus = Mid(strtoparse, readpos, 1)
1032 If oStory.AvailabilityStatus = "P" Then
1034     Err.Raise 2, , "Ignore Error: P(Test)
1036     End If
1040 End If
1050 endpos = InStr(readpos, strtoparse, "CONTACT", vbBinaryCompare)
1060 readpos = endpos + 9
1070 endpos = InStr(readpos, strtoparse, "/")
1080 oStory.Contact = Mid(strtoparse, readpos, endpos - readpos)
1090 oStory.Contact = Replace(oStory.Contact, "/", "'" ' ' ' ' ' ' ' vbTextCompare)
1100 'MsgBox "Contact" & oStory.Contact
1105 ' 9/01/99 Everything is good with the specifications up to this point. ......
1110 readpos = endpos + 3
1120 If Mid$(strtoparse, readpos, 1) <> Chr$(CR) And
1130 Mid$(strtoparse, readpos, 1) <> " " Then
1135 endpos = InStr(readpos, strtoparse, "/")
1140 readpos = endpos + 1
1150 endpos = InStr(readpos, strtoparse, "/")
1155 'Changed If... to While because we don't know how many
1160 While Mid(strtoparse, endpos, 2) = "/"
1170     endpos = endpos + 2
1180     endpos = InStr(endpos, strtoparse, "/")
1190 Wend
1200 oStory.LinkedURL = Mid(strtoparse, readpos, endpos - readpos)
1210 oStory.LinkedURL = Replace(oStory.LinkedURL, "'", "'", ' ' ' ' ' vbTextCompare)
1220 'MsgBox "LinkedURL=" & oStory.LinkedURL
1230 End If
1235 'Look good up to here.....
1240 endpos = InStr(readpos, strtoparse, "CO:")
1250 readpos = endpos + 3
1260 endpos = InStr(readpos, strtoparse, vbCrLf)
readpos, endpos = readpos)
1280 oStory.SignificantCompany = """
Replace(oStory.SignificantCompany, "\", "\"", ", ",
1290 'MsgBox "SigCo=" & oStory.SignificantCompany
endpos = InStr(readpos, strtoparse, "ST:")
1300 readpos = endpos + 3
1310 endpos = InStr(readpos, strtoparse, vbCrLf)
1320 oStory.SignificantState = Mid(strtoparse, readpos, endpos - readpos)
1330 oStory.SignificantState = """
Replace(oStory.SignificantState, "\", "\"", ", ",
1340 'MsgBox "SigStory=" & oStory.SignificantState
1350 endpos = InStr(readpos, strtoparse, "IN:")
1355 readpos = endpos + 3
1360 endpos = InStr(readpos, strtoparse, vbCrLf)
1370 oStory.SignificantIndustryCode = Mid(strtoparse, readpos, endpos - readpos)
1380 'MsgBox "SigIndustry=" &
1390 oStory.SignificantIndustryCode
1395 endpos = InStr(readpos, strtoparse, "SU:")
1400 readpos = endpos + 3
1405 endpos = InStr(readpos, strtoparse, vbCrLf)
1410 oStory.SignificantSubjectCode = Mid(strtoparse, readpos, endpos - readpos)
1420 'MsgBox "SigSubCode=" &
1430 oStory.SignificantSubjectCode
1435 endpos = InStr(readpos, strtoparse, Chr$(ETX))
1440 endpos = endpos + 3
1445 endpos = endpos
1450 endpos = InStr(readpos, strtoparse, vbCrLf)
1455 oStory.VDCEditorInitials = Mid(strtoparse, readpos, endpos - readpos)
1460 'MsgBox "Initials=" & oStory.VDCEditorInitials
1470 oStory.VDCEditorInitials = Mid(strtoparse, readpos, endpos - readpos)
1480 'MsgBox "PRSFilenumber=" & oStory.PRSFileNumber
1490 endpos = InStr(readpos, strtoparse, vbCrLf)
1500 endpos = endpos + 2
1505 endpos = endpos
1510 endpos = InStr(readpos, strtoparse, "--")
1515 readpos = endpos + 3
1520 oStory.PRSFileNumber = Mid(strtoparse, readpos, endpos - readpos)
1525 'MsgBox "PRSFilenumber=" & oStory.PRSFileNumber
1530 endpos = InStr(readpos, strtoparse, vbCrLf)
1535 endpos = endpos + 2
1540 endpos = endpos
1545 endpos = InStr(readpos, strtoparse, "")
1550 readpos = endpos + 1
1555 endpos = InStr(readpos, strtoparse, "")
1560 oStory.PRSDate = Mid(strtoparse, readpos, endpos - readpos)
1565 'MsgBox "PRSFilenumber=" & oStory.PRSDate
1570 readpos = endpos + 1
1575 endpos = InStr(readpos, strtoparse, "")
1580 readpos = endpos + 1
1585 endpos = InStr(readpos, strtoparse, "")
1590 oStory.PRSDate = Mid(strtoparse, readpos, endpos - readpos)
1595 'MsgBox "PRSFilenumber=" & oStory.PRSDate
1600 endpos = InStr(readpos, strtoparse, ")"
oStory.PRSTime = Mid(strtoparse, readpos, endpos - readpos)
'MsgBox "PRSTime=" & oStory.PRSTime
readpos = endpos + 1
endpos = InStr(readpos, strtoparse, " ")
endpos - readpos)
'MsgBox "TimeZone=" & oStory.PRSTimeZone
readpos = endpos + 1
endpos = InStr(readpos, strtoparse, Chr$(ETX))
endpos - readpos)
'MsgBox "PRSURL=" & oStory.PRSURL
'MsgBox "Before Call"
TrimStory oStory
Open "c:\debuganpa.txt" For Append As #filename
Write #filename, oStory.AvailabilityStatus
Write #filename, oStory.VDCEditorInitials
Write #filename, oStory.Mod_dt
Close #filename
Set oServerManager = CreateObject("NewswireManager.cDBANPA")
strjunk = oServerManager.createANPA(oStory)
'MsgBox strjunk
Set oStory = Nothing
'MsgBox "After return"
Set oServerManager = Nothing
oFeedParser.NotifyMe (False)
Exit_Parse:
   Exit Sub
Err_Parse:
   If Err = 990 Then
   sRetCode = "Line Number: " & Err & " 
   oServerEvent.ReportEvent
   scomputername, "FeedParser", "modParse", "Parse", sRetCode
   Resume Exit_Parse
End If
Open "C:\story.txt" For Append As #filename
Write #filename, strtoparse
Write #filename, vbCrLf
Close #filename
sRetCode = "Line Number: " & Err & " 
Err: " & Err.Number & " Error: " & Err.Description
oServerEvent.ReportEvent
scomputername, "FeedParser", "modParse", "Parse", sRetCode
Set oStory = Nothing
oFeedParser.NotifyMe (True)
Public Sub TrimStory(ByRef oANPA)
On Error GoTo Err_TrimStory

20   If IsNull(oANPA.Addressing) Or
21       Len(Trim(oANPA.Addressing)) = 0 Then
22       Else
23          oANPA.Addressing = Trim(oANPA.Addressing)
24       End If
25
26   If IsNull(oANPA.AvailabilityDate) Or
27       Len(Trim(oANPA.AvailabilityDate)) = 0 Then
28       Else
29          oANPA.AvailabilityDate =
30          Trim(oANPA.AvailabilityDate)
31       End If
32
33   ' If IsNull(oANPA.AvailabilityStatus) Or
34       Len(Trim(oANPA.AvailabilityStatus)) = 0 Then
35       oANPA.AvailabilityStatus = "U"
36   Else
37
38       oANPA.AvailabilityStatus =
39       Trim(oANPA.AvailabilityStatus)
40   End If
41
42   ' If IsNull(oANPA.ENQStatus) Or
43       Len(Trim(oANPA.ENQStatus)) = 0 Then
44       oANPA.ENQStatus = "R"
45   Else
46
47       oANPA.ENQStatus = Trim(oANPA.ENQStatus)
48   End If
49
50   If IsNull(oANPA.BureauBanner) Or
51       Len(Trim(oANPA.BureauBanner)) = 0 Then
52       Else
53
54       oANPA.BureauBanner =
55       Trim(oANPA.BureauBanner)
56       End If
57
58   If IsNull(oANPA.Category) Or
59       Len(Trim(oANPA.Category)) = 0 Then
60       Else
61
62       oANPA.Category = Trim(oANPA.Category)
63       End If
64
65   If IsNull(oANPA.Contact) Or
66       Len(Trim(oANPA.Contact)) = 0 Then
67       Else
68
69       oANPA.Contact = Trim(oANPA.Contact)
70       End If
71
72   If IsNull(oANPA.FiveDigit) Or
73       Len(Trim(oANPA.FiveDigit)) = 0 Then
74
Else
    oANPA.FiveDigit = Trim(oANPA.FiveDigit)
End If
If IsNull(oANPA.Headline) Or Len(Trim(oANPA.Headline)) = 0 Then
    Else
    oANPA.Headline = Trim(oANPA.Headline)
End If
If IsNull(oANPA.IndustryCode) Or Len(Trim(oANPA.IndustryCode)) = 0 Then
    Else
    oANPA.IndustryCode = Trim(oANPA.IndustryCode)
End If
If IsNull(oANPA.LinkedURL) Or Len(Trim(oANPA.LinkedURL)) = 0 Then
    Else
    oANPA.LinkedURL = Trim(oANPA.LinkedURL)
End If
oANPA.Add_dt = Now
'8/2/99 Tim added line.
oANPA.Add_dt = ""
oANPA.Mod_dt = ""
If IsNull(oANPA.Priority) Or Len(Trim(oANPA.Priority)) = 0 Then
    'Set in table default value of priority to 3 or read from ini file and assign here.
    Else
    oANPA.Priority = Trim(oANPA.Priority)
End If
If IsNull(oANPA.PRSEditorInitials) Or Len(Trim(oANPA.PRSEditorInitials)) = 0 Then
    Else
    oANPA.PRSEditorInitials = Trim(oANPA.PRSEditorInitials)
End If
If IsNull(oANPA.PRSFileNumber) Or Len(Trim(oANPA.PRSFileNumber)) = 0 Then
    Else
    oANPA.PRSFileNumber = Trim(oANPA.PRSFileNumber)
End If
If IsNull(oANPA.PRSTime) Or Len(Trim(oANPA.PRSTime)) = 0 Then
    Else
    oANPA.PRSTime = Trim(oANPA.PRSTime)
End If
If IsNull(oANPA.PRSTimeZone) Or Len(Trim(oANPA.PRSTimeZone)) = 0 Then
    Else
    oANPA.PRSTimeZone = Trim(oANPA.PRSTimeZone)
680   End If
690   If IsNull(oANPA.PRSURL) Or
695    Len(Trim(oANPA.PRSURL)) = 0 Then
700     oANPA.PRSURL = Trim(oANPA.PRSURL)
705   End If
710   If IsNull(oANPA.Reference) Or
715    Len(Trim(oANPA.Reference)) = 0 Then
720     oANPA.Reference = Trim(oANPA.Reference)
725   End If
730   If IsNull(oANPA.ServiceLevelDes) Or
735    Len(Trim(oANPA.ServiceLevelDes)) = 0 Then
740     oANPA.ServiceLevelDes =
745    Trim(oANPA.ServiceLevelDes)
750   End If
755   If IsNull(oANPA.SignificantCompany) Or
760    Len(Trim(oANPA.SignificantCompany)) = 0 Then
765     oANPA.SignificantCompany =
770    Trim(oANPA.SignificantCompany)
775   End If
780   If IsNull(oANPA.SignificantIndustryCode) Or
785    Len(Trim(oANPA.SignificantIndustryCode)) = 0 Then
790     oANPA.SignificantIndustryCode =
795    Trim(oANPA.SignificantIndustryCode)
800   End If
805   If IsNull(oANPA.SignificantState) Or
810    Len(Trim(oANPA.SignificantState)) = 0 Then
815     oANPA.SignificantState =
820    Trim(oANPA.SignificantState)
825   End If
830   If IsNull(oANPA.SignificantSubjectCode) Or
835    Len(Trim(oANPA.SignificantSubjectCode)) = 0 Then
840     oANPA.SignificantSubjectCode =
845    Trim(oANPA.SignificantSubjectCode)
850   End If
855   If IsNull(oANPA.Source) Or
860    Len(Trim(oANPA.Source)) = 0 Then
865     oANPA.Source = Trim(oANPA.Source)
870   End If
875   If IsNull(oANPA.StockSymbols) Or
880    Len(Trim(oANPA.StockSymbols)) = 0 Then
885     oANPA.StockSymbols =
890    Trim(oANPA.StockSymbols)
895   End If
1060  If IsNull(oANPA.StoryKeyword) Or
1070    Len(Trim(oANPA.StoryKeyword)) = 0 Then
1080      Else
1090      oANPA.StoryKeyword =
1100    Trim(oANPA.StoryKeyword)
1110  End If
1120  If IsNull(oANPA.SubjectCode) Or
1130    Len(Trim(oANPA.SubjectCode)) = 0 Then
1140      Else
1150      oANPA.SubjectCode = Trim(oANPA.SubjectCode)
1160    End If
1170  If IsNull(oANPA.TransmissionDate) Or
1180    Len(Trim(oANPA.TransmissionDate)) = 0 Then
1190      Else
1200    oANPA.TransmissionDate = Date
1210  End If
1220    oANPA.VDCEditorInitials = ""
1230  If IsNull(oANPA.Version) Or
1240    Len(Trim(oANPA.Version)) = 0 Then
1250      Else
1260    oANPA.Version = Trim(oANPA.Version)
1270  End If
1280  Exit Sub
1290  End Sub
1300
1310  cParse.cls

1320 Option Explicit
1330  Event Notify(flag As Boolean)
1340
1350  Public Sub NotifyMe(flag As Boolean)
1360    On Error GoTo Err_NotifyMe
1370      RaiseEvent Notify(flag)
1380    Exit Sub
1390    Err_NotifyMe:
1400      sRetCode = "Line Number: " & Erl & " Err: " &
1410        Err.Number & " Error: " & Err.Description
1420      oServerEvent.ReportEvent strcomputername,
1430        "FeedParser", "cParse", "NotifyMe", sRetCode
1440    End Sub
1450
1460  Public Sub ParseString(strtoparse As String)
1470    On Error GoTo Err_ParseString
1480      Call Parse(Me, strtoparse)
1490    Exit Sub
1500  Err_ParseString:
sRetCode = "Line Number: " & Err & " Err: " &
Err.Number & " Error: " & Err.Description
oServerEvent.ReportEvent strcomputername,
"FeedParser", "cParse", "ParseString", sRetCode
End Sub

Public Sub SendObject()
    Call SendObjects
End Sub

Private Sub Class_Initialize()
    Dim rc As Integer
    Set oServerEvent =
    CreateObject("NewsWireManager.cEventLog")
    rc = GetComputerName(strcomputername,
Len(strcomputername))
End Sub

Private Sub Class_Terminate()
    Set oServerEvent = Nothing
End Sub

{Copyright 1999 VISUAL DATA CORPORATION}

The feed parser 44 parses through the complete press
release story to identify individual data elements, such as
headline text, story text, stock symbol text, industry code
text, etc., assigns these data elements to data fields in
the press release object 46, and sends the press release
object 46 to the server 50. The server 50 stores each of the
fields of the press release object 46 into a record in the
database 12 and adds the release to the editor queue 52. In
addition, the server 50 keeps track of the status of all the
press releases as they flow through the system, updating the
statuses, e.g. unedited or edited, in the database 12 as
they change. Once a press release enters the editor queue 52
it is sent by the server 50 to each of the editor computers
14, 16, & 18.

Newswire1.dll
------ cANPA.cls
------ cPermission.cls

Option Explicit

Private tANPAID As Integer
Private tEditedType As String
Private tServiceLevelDes As String
Private tVDCEditorInitials As String
Private tFaveDigit As String
Private tPriority As String
Private tCategory As String
Private tStoryKeyword As String
Private tVersion As String
Private tReference As String
Private tTransmissionDate As String
Private tBureauBanner As String
Private tStockSymbols As String
Private tIndustryCode As String
Private tSubjectCode As String
Private tAddressing As String
Private tHeadline As String
Private tParagraphs As String
Private tSource As String
Private tAvailabilityDate As Date
Private tAvailabilityStatus As String
Private tContact As String
Private tLinkedURL As String
Private tSignificantCompany As String
Private tSignificantState As String
Private tSignificantIndustryCode As String
Private tSignificantSubjectCode As String
Private tPRESEditorInitials As String
Private tPRSFileNumber As String
Private tPRSDate As Date
Private tPRSTime As String
Private tPRSTimeZone As String
Private tPRSURL As String
Private tEditedMod_dt As String
Private tAdd_dt As String
Private tMod_dt As String
Private tLockStatus As String
Private tENQStatus As String
Private tComputerName As String
Private tPreview As Boolean

Public Property Get ServiceLevelDes() As String
    ServiceLevelDes = tServiceLevelDes
End Property

Public Property Let ServiceLevelDes(ByVal vNewValue As String)
    tServiceLevelDes = vNewValue
End Property

Public Property Get ANPAID() As Integer
    ANPAID = tANPAID
End Property
Public Property Let ANPAID(ByVal vNewValue As Integer)
    tANPAID = vNewValue
End Property

Public Property Get FiveDigit() As Variant
    FiveDigit = tFiveDigit
End Property

Public Property Let FiveDigit(ByVal vNewValue As Variant)
    tFiveDigit = vNewValue
End Property

Public Property Get Priority() As Variant
    Priority = tPriority
End Property

Public Property Get EditedType() As Variant
    EditedType = tEditedType
End Property

Public Property Let Priority(ByVal vNewValue As Variant)
    tPriority = vNewValue
End Property

Public Property Let EditedType(ByVal vNewValue As Variant)
    tEditedType = vNewValue
End Property

Public Property Get Category() As Variant
    Category = tCategory
End Property

Public Property Let Category(ByVal vNewValue As Variant)
    tCategory = vNewValue
End Property

Public Property Get StoryKeyword() As Variant
    StoryKeyword = tStoryKeyword
End Property

Public Property Let StoryKeyword(ByVal vNewValue As Variant)
    tStoryKeyword = vNewValue
End Property

Public Property Get Version() As Variant
    Version = tVersion
End Property
Public Property Let Version(ByVal vNewValue As Variant)  
    tVersion = vNewValue  
End Property

Public Property Get Reference() As Variant  
    Reference = tReference  
End Property

Public Property Let Reference(ByVal vNewValue As Variant)  
    tReference = vNewValue  
End Property

Public Property Get TransmissionDate() As Variant  
    TransmissionDate = tTransmissionDate  
End Property

Public Property Let TransmissionDate(ByVal vNewValue As Variant)  
    tTransmissionDate = vNewValue  
End Property

Public Property Get BureauBanner() As Variant  
    BureauBanner = tBureauBanner  
End Property

Public Property Let BureauBanner(ByVal vNewValue As Variant)  
    tBureauBanner = vNewValue  
End Property

Public Property Get LockStatus() As Variant  
    LockStatus = tLockStatus  
End Property

Public Property Let LockStatus(ByVal vNewValue As Variant)  
    tLockStatus = vNewValue  
End Property

Public Property Get ENQStatus() As Variant  
    ENQStatus = tENQStatus  
End Property

Public Property Let ENQStatus(ByVal vNewValue As Variant)  
    tENQStatus = vNewValue  
End Property

Public Property Get ComputerName() As Variant  
    ComputerName = tComputerName  
End Property
Public Property Let ComputerName(ByVal vNewValue As Variant)
    tComputerName = vNewValue
End Property

Public Property Get StockSymbols() As Variant
    StockSymbols = tStockSymbols
End Property

Public Property Let StockSymbols(ByVal vNewValue As Variant)
    tStockSymbols = vNewValue
End Property

Public Property Get IndustryCode() As Variant
    IndustryCode = tIndustryCode
End Property

Public Property Let IndustryCode(ByVal vNewValue As Variant)
    tIndustryCode = vNewValue
End Property

Public Property Get SubjectCode() As Variant
    SubjectCode = tSubjectCode
End Property

Public Property Let SubjectCode(ByVal vNewValue As Variant)
    tSubjectCode = vNewValue
End Property

Public Property Get Addressing() As Variant
    Addressing = tAddressing
End Property

Public Property Let Addressing(ByVal vNewValue As Variant)
    tAddressing = vNewValue
End Property

Public Property Get Headline() As Variant
    Headline = tHeadline
End Property

Public Property Let Headline(ByVal vNewValue As Variant)
    tHeadline = vNewValue
End Property

Public Property Get Paragraphs() As Variant
    Paragraphs = tParagraphs
End Property

Public Property Let Paragraphs(ByVal vNewValue As Variant)
    tParagraphs = vNewValue
End Property

Public Property Get Source() As Variant
    Source = tSource
End Property

Public Property Let Source(ByVal vNewValue As Variant)
    tSource = vNewValue
End Property

Public Property Get AvailabilityDate() As Variant
    AvailabilityDate = tAvailabilityDate
End Property

Public Property Let AvailabilityDate(ByVal vNewValue As Variant)
    tAvailabilityDate = vNewValue
End Property

Public Property Get AvailabilityStatus() As Variant
    AvailabilityStatus = tAvailabilityStatus
End Property

Public Property Let AvailabilityStatus(ByVal vNewValue As Variant)
    tAvailabilityStatus = vNewValue
End Property

Public Property Get Contact() As Variant
    Contact = tContact
End Property

Public Property Let Contact(ByVal vNewValue As Variant)
    tContact = vNewValue
End Property

Public Property Get LinkedURL() As Variant
    LinkedURL = tLinkedURL
End Property

Public Property Let LinkedURL(ByVal vNewValue As Variant)
    tLinkedURL = vNewValue
End Property

Public Property Get SignificantCompany() As Variant
    SignificantCompany = tSignificantCompany
End Property

Public Property Let SignificantCompany(ByVal vNewValue As Variant)
    tSignificantCompany = vNewValue
End Property

Public Property Get SignificantState() As Variant
    SignificantState = tSignificantState
End Property

Public Property Let SignificantState(ByVal vNewValue As Variant)
    tSignificantState = vNewValue
End Property

Public Property Get SignificantIndustryCode() As Variant
    SignificantIndustryCode = tSignificantIndustryCode
End Property

Public Property Let SignificantIndustryCode(ByVal vNewValue As Variant)
    tSignificantIndustryCode = vNewValue
End Property

Public Property Get SignificantSubjectCode() As Variant
    SignificantSubjectCode = tSignificantSubjectCode
End Property

Public Property Let SignificantSubjectCode(ByVal vNewValue As Variant)
    tSignificantSubjectCode = vNewValue
End Property

Public Property Get PRSEditorInitials() As Variant
    PRSEditorInitials = tPRSEditorInitials
End Property

Public Property Let PRSEditorInitials(ByVal vNewValue As Variant)
    tPRSEditorInitials = vNewValue
End Property

Public Property Get PRSFileNumber() As Variant
    PRSFileNumber = tPRSFileNumber
End Property

Public Property Let PRSFileNumber(ByVal vNewValue As Variant)
    tPRSFileNumber = vNewValue
End Property
Public Property Get PRSDate() As Variant
    PRSDate = tPRSDate
End Property

Public Property Let PRSDate(ByVal vNewValue As Variant)
    tPRSDate = vNewValue
End Property

Public Property Get PRSTime() As Variant
    PRSTime = tPRSTime
End Property

Public Property Let PRSTime(ByVal vNewValue As Variant)
    tPRSTime = vNewValue
End Property

Public Property Get PRSTimeZone() As Variant
    PRSTimeZone = tPRSTimeZone
End Property

Public Property Let PRSTimeZone(ByVal vNewValue As Variant)
    tPRSTimeZone = vNewValue
End Property

Public Property Get PRSURL() As Variant
    PRSURL = tPRSURL
End Property

Public Property Let PRSURL(ByVal vNewValue As Variant)
    tPRSURL = vNewValue
End Property

Public Property Get EditedMod_dt() As String
    EditedMod_dt = tEditedMod_dt
End Property

Public Property Let EditedMod_dt(ByVal vNewValue As String)
    tEditedMod_dt = vNewValue
End Property

Public Property Get Mod_dt() As String
    Mod_dt = tMod_dt
End Property

Public Property Let Mod_dt(ByVal vNewValue As String)
    tMod_dt = vNewValue
End Property

Public Property Get Add_dt() As String
    Add_dt = tAdd_dt
End Property

Public Property Let Add_dt(ByVal vNewValue As String)
    tAdd_dt = vNewValue
End Property

Public Property Get VDCEditorInitials() As Variant
    VDCEditorInitials = tVDCEditorInitials
End Property

Public Property Let VDCEditorInitials(ByVal vNewValue As Variant)
    tVDCEditorInitials = vNewValue
End Property

Public Property Get Preview() As Variant
    Preview = tPreview
End Property

Public Property Let Preview(ByVal vNewValue As Variant)
    tPreview = vNewValue
End Property

cPermission.cls

Private iEMPID As Integer
Private sUserID As String
Private sPassword As String
Private sSysRole As String
Private sEditorRole As String
Private sNarratorRole As String
Private sViewerRole As String
Private dModdt As Date
Private sVDCCUserInitials As String

Public Property Get UserID() As Variant
    UserID = sUserID
End Property

Public Property Let UserID(ByVal vNewValue As Variant)
    sUserID = vNewValue
End Property

Public Property Get Password() As Variant
    Password = sPassword
End Property

Public Property Let Password(ByVal vNewValue As Variant)
    sPassword = vNewValue
End Property

Public Property Get SysRole() As Variant
    SysRole = sSysRole
End Property

Public Property Let SysRole(ByVal vNewValue As Variant)
    sSysRole = vNewValue
End Property

Public Property Get EditorRole() As Variant
    EditorRole = sEditorRole
End Property

Public Property Let EditorRole(ByVal vNewValue As Variant)
    sEditorRole = vNewValue
End Property

Public Property Get NarratorRole() As Variant
    NarratorRole = sNarratorRole
End Property

Public Property Let NarratorRole(ByVal vNewValue As Variant)
    sNarratorRole = vNewValue
End Property

Public Property Get ViewerRole() As Variant
    ViewerRole = sViewerRole
End Property

Public Property Let ViewerRole(ByVal vNewValue As Variant)
    sViewerRole = vNewValue
End Property

Public Property Get Moddt() As Variant
    Moddt = dModdt
End Property

Public Property Let Moddt(ByVal vNewValue As Variant)
    dModdt = vNewValue
End Property

Public Property Get EMPID() As Variant
    EMPID = iEMPID
End Property

Public Property Let EMPID(ByVal vNewValue As Variant)
    iEMPID = vNewValue
End Property

Public Property Get VDCUserInitials() As Variant
    VDCUserInitials = sVDCUserInitials
End Property

Public Property Let VDCUserInitials(ByVal vNewValue As Variant)
    sVDCUserInitials = vNewValue
End Property

(Copyright 1999 VISUAL DATA CORPORATION)

Within each editor computer 14, 16, & 18 is an editor client 56, 58, & 60 software module, which is the interface between a user, i.e. editor, and the server 50. Upon receipt of the press release from the server 50, the editor client 56 adds the press release to a list box which is displayed on the editor computer 14, 16 & 18. The list box displays all of the press releases which are currently in the editor queue 52. The list box enables the user to select a press release and execute one or more operations, such as edit, assign priority, save, delete, skip, or cancel. When the user executes an operation, the editor client 56, 58, & 60 makes the appropriate method call to the server 50 to perform the operation. When the operation is completed, the server 50 informs the editor clients 56, 58, & 60 so that it can update the information displayed to the user. The user may edit the content of a press release narrative and may assign a priority code and preview flag that will allow important stories to be read by the narrator in a timely manner.

The narrator queue 54 is managed by the server 50. After a press release has been edited and saved by a user on one of the editor computers 14, 16, or 18, the server 50 adds the press release to the narrator queue 54, ordering the press releases in the queue according to the priority codes assigned by the editor.

Within the narrator computer 20 is the narrator client 62 software module and the media encoder 64. The narrator client 62 is the interface between the user, i.e. narrator, the server 50 and the media server 22. The narrator requests
the next press release When the narrator is ready to read
the next press release, the narrator enters the request at
the narrator computer 20 interface by such means as a
keystroke, button push, or by clicking on an icon. The
narrator client 62 responds by instructing the server 50 to
send the next press release to the display of the narrator
computer 20 and instructing the media encoder 64 to send a
command and text string through the live media stream. The
live media stream is the audio data that is generated by the
narrator computer 20 as the narrator reads the press release
into an audio input device. The content of the text string
command depends on the command type, such as HEADLINE,
MARKET UPDATE, END OF DAY, and PROMO. For the HEADLINE
command type the text includes the stock symbol, industry
code, the headline, and the unique id of the press release
record in the database 12. The live media stream with the
encoded text string is sent to the media server 22.

On the web server 26 there are Active Server Pages 66,
which is the software that manages user registration, user
preferences, and user sign-in. Registration is required to
gain access to the service. Once registered and signed in,
the user can create their preferences by selecting up to ten
industry codes and twenty-five stock symbols to have on
their watch list. These preferences are stored in the user’s
record in the database 12 and stored in “cookies” on the
user computer 24. A “cookie” is a text file that contains
the user’s preferences. On subsequent sign-ins the
preferences are retrieved from the database and stored in
the “cookies”, thus enabling the user to sign in at
different times from different computers.

After successfully signing in to TheFirstNews web site,
the user is provide buttons to start the TheFirstNews
Player. The TheFirstNews Player 68 software with embedded
media player 70 is downloaded to the user computer 24. When
the user starts the TheFirstNews Player 68, the software
reads the preferences from the "cookies", stores them in
local variables on the user computer 24, starts the media
player 70, and establishes a communication link with the
live media stream from the media server 22. When the script
command and text string that was encoded into the live media
stream is received by the media player 70, the player calls
a function that determines what type of command has been
received, such as HEADLINE, MARKET UPDATE, END OF DAY, or
PROMO. When the HEADLINE command is received the text string
is decoded to get the id, stock symbol, industry code, and
headline text for the live media stream that is about to be
played by the media player 70. The stock symbol and industry
code are compared to the preferences set by the user. If
there is a match, then an instruction is sent to the media
player 70 to insure that the volume is turned on if the user
has it muted. The stock symbol and headline are also
displayed highlighted in a scrolling table. The headline
text is also an active URL (Uniform Resource Locator) link
to the full text of the press release stored in the database
12.

if (document.images) {
    var aa = new Image()
    aa.src = "../images/aa.gif"
    var az = new Image()
    az.src = "../images/az.gif"
    var pa = new Image()
    pa.src = "../images/pa.gif"
    var pz = new Image()
    pz.src = "../images/pz.gif"
    var ra = new Image()
    ra.src = "../images/ra.gif"
    var rz = new Image()
    rz.src = "../images/rz.gif"
    var ua = new Image()
    ua.src = "../images/ua.gif"
    var uz = new Image()
    uz.src = "../images/uz.gif"
    var ba = new Image()
    ba.src = "../images/ba.gif"
    var bz = new Image()
bz.src = "../images/bz.gif"
var ca = new Image()
ca.src = "../images/ca.gif"
var cz = new Image()
cz.src = "../images/cz.gif"
var da = new Image()
da.src = "../images/da.gif"
var dz = new Image()
dz.src = "../images/dz.gif"
var ea = new Image()
ea.src = "../images/ea.gif"
var ez = new Image()
ez.src = "../images/ez.gif"
var fa = new Image()
fa.src = "../images/fa.gif"
var fz = new Image()
fz.src = "../images/fz.gif"
var ga = new Image()
ga.src = "../images/ga.gif"
var gz = new Image()
gz.src = "../images/gz.gif"
var ha = new Image()
ha.src = "../images/ha.gif"
var hz = new Image()
hz.src = "../images/hz.gif"
var qa = new Image()
qa.src = "../images/qa.gif"
var qz = new Image()
qz.src = "../images/qz.gif"
}
function act(imgName) {
    if (document.images)
        document[imgName].src = eval(imgName + 'z.src')
}
function inact(imgName) {
    if (document.images)
        eval(imgName + 'a.src')
}

function GetCookieValue(cookieName)
{
    var pos = cookieStr.indexOf(cookieName + "=");
    if (pos !== -1)
    {
        var start = pos + cookieName.length + 1;
        var end = cookieStr.indexOf(";", start);
        if (end == -1) end = cookieStr.length;
        var value = cookieStr.substring(start, end);
value = unescape(value);
return value;
}
else
return 0;)

function myVoid() {;} //do nothing

var win1;
function launch()
{
// if (typeof(win1) == 'undefined')
// {
win1=window.open('', 'client', 'width=470,height=400,location=
no,menubar=no,resizable=yes,scrollbars=no,status=no,titlebar
=no,toolbar=no,screenX=0,screenY=0,top=0,left=0');
win1.location.href = "player/index1.asp";
if (win1.opener == null) win1.opener = self;
// }
else
// win1.focus();
}

var win2;
function launchdemo()
{
// if (typeof(win2) == 'undefined')
// {
win2=window.open('', 'client', 'width=470,height=400,location=
no,menubar=no,resizable=yes,scrollbars=no,status=no,titlebar
=no,toolbar=no,screenX=0,screenY=0,top=0,left=0');
win2.location.href = "demo/index1.asp";
if (win2.opener == null) win2.opener = self;
// }
else
// win2.focus();

(Copyright 1999 VISUAL DATA CORPORATION)

If there is no match with the user's preferences the volume of the player is maintained at the user's setting and the headline is displayed in normal text. When the other commands and text strings are received an instruction is sent to the media player 70 to insure that the volume is turned on if the user has it muted and a message is displayed to the user. The user can adjust the volume level of the media player 70 by using the up and down arrows on the player.
It is to be understood that while I have illustrated and described certain forms of my invention, it is not to be limited to the specific forms or arrangement of parts herein described and shown. It will be apparent to those skilled in the art that various changes may be made without departing from the scope of the invention and the invention is not to be considered limited to what is shown in the drawings and described in the specification.
CLAIMS

1. A method for manipulating a live audio media stream from a global computer network by an end user having a terminal coupled to said global computer network, comprising:
   accessing a text feed by a computer network and parsing said text feed into individual components;
   storing said individual components on said computer network and at least one server computer, providing an editor queue and a narrator queue;
   sending said individual components to an editor computer and enabling an editor to assign priority codes;
   managing said narrator queue and ordering media releases by priority codes;
   maintaining a record of all said media releases and updating said computer network upon change of status;
   receiving and maintaining a record of user preferences related to said individual components;
   sending the next said media release as ordered in the narrator queue to a narrator computer, enabling release to be read by a narrator and generating a live audio media stream of said release;
   calling a media encoder by narrator client software to send a command text string through the said live audio media stream;
   launching a media player and receiving said media stream;
   decoding said text string and comparing to said preferences set by the user, upon matching of said preferences and turning up the volume of said medial player;
   displaying of individual components meeting user preferences from said media release.
Claim 2. The method for manipulating a live audio media stream according to Claim 1 including the step of providing an active URL link to the full text of said media release.

Claim 3. The method for manipulating a live audio media stream according to Claim 1 including the step of returning of said volume to an initial state if user preference are not matched.

Claim 4. The method for manipulating a live audio media stream according to Claim 1 wherein said commands are head lines, market updates, end of day, and commercials.

Claim 5. The method for manipulating a live audio media stream according to Claim 1 wherein said components are stock symbols, industry codes, and headline text.

Claim 6. The method for manipulating a live audio media stream according to Claim 1 wherein said display of said individual components are in a scrolling table.

Claim 7. The method for manipulating a live audio media stream according to Claim 1 including the step of populating said narrator queue after an editor has edited and saved a media release to said editor computer.

Claim 8. The method for manipulating a live audio media stream according to Claim 1 including a data base for recording of user preferences.

Claim 9. The method for manipulating a live audio media stream according to Claim 8 wherein said user preferences are up to about 10 industry codes and up to about 25 stock symbols.
Claim 10. The method for manipulating a live audio media stream according to Claim 1 wherein said display is highlighted.

Claim 11. A method for manipulating a live audio media stream from a global computer network by an end user having a terminal coupled to said global computer network, comprising:

- accessing a news wire text feed by a computer network and parsing said news wire text feed into individual components;
- storing said individual components on said computer network and at least one server computer, providing an editor queue and a narrator queue;
- sending said individual components to an editor computer and enabling an editor to assign priority codes;
- managing said narrator queue and ordering media releases by priority codes;
- maintaining a record of all said media releases and updating said computer network upon change of status;
- receiving and maintaining a record of user preferences related to said individual components;
- sending the next said media release as ordered in the narrator queue to a narrator computer, enabling release to be read by a narrator and generating a live audio media stream of said release;
- calling a media encoder by narrator client software to send a command and text string through the said live audio media stream;
- launching a media player and receiving said media stream;
- decoding said text string and comparing to said preferences set by the user, upon matching of said preferences and turning up the volume of said medial player;
providing a highlight and scrolling display of
individual components meeting user preferences from said
media release; and
returning of said volume to an initial state if user
preference are not matched.

Claim 12. The method for manipulating a live audio
media stream according to Claim 11 wherein said commands are
head lines, market updates, end of day, and commercials.

Claim 13. The method for manipulating a live audio
media stream according to Claim 11 wherein said components
are stock symbols, industry codes, and headline text.

Claim 14. The method for manipulating a live audio
media stream according to Claim 11 including the step of
populating said narrator queue after an editor has edited
and saved a media release to said editor computer.

Claim 15. The method for manipulating a live audio
media stream according to Claim 11 including a web server
for recording of user preferences.

Claim 16. The method for manipulating a live audio
media stream according to Claim 15 wherein said user
preferences are up to about 10 industry codes and up to
about 25 stock symbols.

Claim 17. The method for manipulating a live audio
media stream according to Claim 10 including the step
displaying said individual components in normal text in said
display if user preference are not matched.