Textware™ SOLUTIONS

58 Lexington Street
Burlington MA 01803-4005
781 272 3200
781 272 1432 fax

The API for Instant Text

An Application Programming Interface for Applications using Instant Text

Textware Solutions

The API for Instant Text

How to Use Instant Text as the Abbreviation Engine for your Applications

Introduction

The Application Programming Interface (API) for Instant Text allows you to use Instant Text as an abbreviation expansion engine for your applications. The API is supplied as a DLL (dynamic link library), which can be called by your application to control all aspects of Instant Text. For example, you can call functions and procedures of the DLL to do the following:

- Start Instant Text, minimize it, and restore it.
- Dimension the advisories according to the needs of you application.
- Open glossaries and dynamically select the current glossary according to the field in which text is currently being entered into your application.
- Select the field, the editor, or the memo which is to receive text generated by Instant Text..

For Pen Computer applications, this allows Instant Text and the Fitaly on-screen keyboard to be integrated with your application. Input can be achieved with the Fitaly keyboard and you can use the pen in your application to generate text by abbreviations which depend on the context: the current part of your application.

For Desktop Applications, this also allows development of an integration level similar to that offered for word processors such as MS Word or WordPerfect.

Contents:

The following subjects are covered in the remaining sections of this document:

Description of the API Functions and Procedures

The following describes the function and procedures provided by the Application Programming Interface.

For each entry, we give the Pascal syntax as well as the C and Visual Basic syntax. (The Visual Basic syntax is omitted in the case of functions returning a string.)

Instant_Text_Running

Pascal Syntax:	**function** `Instant_Text_Running: Boolean;`
C Syntax:	`BOOL Instant_Text_Running ();`
VBA Syntax:	`Function Instant_Text_Running () As Boolean`

Returns True if Instant Text is currently running, False otherwise.

Start_Instant_Text

Pascal Syntax:	**procedure** `Start_Instant_Text;`
C Syntax:	`void Start_Instant_Text ();`
VB Syntax:	`Sub Start_Instant_Text ()`

This procedure starts Instant Text if it is not currently running. If Instant Text is currently running and minimized, the effect is to restore it. Otherwise, if Instant Text is currently running and not minimized, it has no effect.

Start_Instant_Text_Lower

Pascal Syntax:	**procedure** `Start_Instant_Text (Lines: Integer);`
C Syntax:	`void Start_Instant_Text (int Lines);`
VB Syntax:	`Sub Start_Instant_Text (ByVal Lines As Integer)`

This procedure starts Instant Text in lower screen position if it is not currently running. If Instant Text is currently running and minimized, the effect is to restore it. Otherwise, if Instant Text is currently running and not minimized, it has no effect.

If the number of lines specified by the Lines parameter is in the range 1 to 15, Instant Text is started with the specified number of advisory lines.

Minimize_Instant_Text;

Pascal Syntax:	**procedure** `Minimize_Instant_Text;`
C Syntax:	`void Minimize_Instant_Text ();`
VB Syntax:	`Sub Minimize_Instant_Text ()`

This procedure minimizes Instant Text.

This procedure has no effect if Instant Text is not currently running.

Restore_Instant_Text;

Pascal Syntax: **procedure** `Restore_Instant_Text;`
C Syntax: `void Restore_Instant_Text ();`
VB Syntax: `Sub Restore_Instant_Text ()`

This procedure restores Instant Text.

This procedure has no effect if Instant Text is not currently running.

Set_IT_Lower_Layout

Pascal Syntax: **procedure** `Set_IT_Lower_Layout;`
C Syntax: `void Set_IT_Lower_Layout ();`
VB Syntax: `Sub Set_IT_Lower_Layout ()`

This procedure switches Instant Text to the lower layout. The effect is similar to that of the shortcut Alt/ when Instant Text is in full-screen layout.

This procedure has no effect if Instant Text is not currently running.

Set_IT_Full_Screen

Pascal Syntax: **procedure** `Set_IT_Full_Screen;`
C Syntax: `void Set_IT_Full_Screen ();`
VB Syntax: `Sub Set_IT_Full_Screen ()`

This procedure switches Instant Text to full-screen layout. The effect is similar to that of the shortcut Alt/ when Instant Text is in lower layout

This procedure has no effect if Instant Text is not currently running.

Instant_Text_Handle

Pascal Syntax: **function** `Instant_Text_Handle: HWnd;`
C Syntax: `HWND Instant_Text_Handle ();`
VB Syntax: `Function Instant_Text_Handle () As Long`

Returns the handle for Instant Text.

Returns the value zero if Instant Text is not currently running.

Get_IT_Rectangle

Pascal Syntax: **procedure** `Get_IT_Rectangle (`**var**` Rect: TRect);`
C Syntax: `void Get_IT_Rectangle (TRect Rect);`

This procedure returns the screen coordinates of Instant Text into the variable Rect.

This procedure has no effect if Instant Text is not currently running.

Get_Advisory_Lines

Pascal Syntax:	**function** `Get_Advisory_Lines: Integer;`
C Syntax:	`int Get_Advisory_Lines ();`
VB Syntax:	`Function Get_Advisory_Lines () As Integer`

Returns the current number of advisory lines.

Returns a default number of lines equal to 8 if Instant Text is not currently running.

Set_Advisory_Lines

Pascal Syntax:	**procedure** `Set_Advisory_Lines (Lines: Integer);`
C Syntax:	`void Set_Advisory_Lines (int Lines);`
VB Syntax:	`Sub Set_Advisory_Lines (ByVal Lines As Integer)`

This procedure changes the number of advisory lines to the number specified by the Lines parameter.

The specified number must be in the range 1 to 15. If the value is lower than 1 or higher than 15, an error message is issued and the procedure has no other effect.

This procedure has no effect if Instant Text is not currently running.

Set_Start_state

Pascal Syntax:	**procedure** `Set_Start_State;`
C Syntax:	`void Set_Start_State ();`
VB Syntax:	`Sub Set_Start_State ()`

This procedure is used to reset Instant Text's internal text as at the start of a paragraph.

The effect is to allow Instant Text to apply the corresponding spacing and capitalization rules.

This procedure has no effect if Instant Text is not currently running.

Close_Instant_Text

Pascal Syntax:	**procedure** `Close_Instant_Text;`
C Syntax:	`void Close_Instant_Text ();`
VB Syntax:	`Sub Close_Instant_Text ()`

This procedure is used to close Instant Text.

This procedure has no effect if Instant Text is not currently running.

Glossary Handling

The following functions and procedures allow:

- opening, reopening, closing, and activating glossaries
- inquiring about the current glossary
- saving the active glossary list
- viewing the current glossary and adding new entries to the current glossary.

Open_Named_Glossary

Pascal Syntax: `procedure Open_Named_Glossary (Name: PChar);`
C Syntax: `void Open_Named_Glossary (LPSTR Name);`
VB Syntax: `Sub Open_Named_Glossary (ByVal Name As String)`

The effect of this procedure is to open the glossary file specified by the parameter Name.

This parameter must be the full path name of a glossary file. The procedure has no effect if Name is not the name of a file or if the file does not contain a glossary.

This procedure has no effect if Instant Text is not currently running.

Reopen_Current_Glossary

Pascal Syntax: `procedure Reopen_Current_Glossary;`
C Syntax: `void Reopen_Current_Glossary ();`
VB Syntax: `Sub Reopen_Current_Glossary ()`

The effect of this procedure is to reopen the current glossary. The effect is the same as that of closing the current glossary and opening it again at the same place in the glossary list. In particular, andy included glossaries are also reopened.

This procedure has no effect if Instant Text is not currently running.

Close_Current_Glossary

Pascal Syntax: `procedure Close_Current_Glossary;`
C Syntax: `void Close_Current_Glossary ();`
VB Syntax: `Sub Close_Current_Glossary ()`

The effect of this procedure is to close the current glossary.

This procedure has no effect if Instant Text is not currently running.

Activate_Glossary_Shortname

```
Pascal Syntax:   procedure Activate_Glossary_Shortname (Name: Char);
     C Syntax:   void Activate_Glossary_Shortname (LPSTR Name);
    VB Syntax:   Sub Activate_Glossary_Shortname (ByVal Name As String)
```

The effect of this procedure is to make the glossary specified by Name active. Name must designate one of the glossaries of the Glossary List by its short name, for example, "Contract" for the glossary Contract.glo.

This procedure has no effect if Name is not the short name of a glossary of the Glossary List or if Instant Text is not currently running.

Activate_Glossary_Longname

```
Pascal Syntax:   procedure Activate_Glossary_Longname (Name: PChar);
     C Syntax:   void Activate_Glossary_Longname (LPSTR Name);
    VB Syntax:   Sub Activate_Glossary_Longname (ByVal Name As String)
```

The effect of this procedure is to make the glossary specified by Name active. Name must designate the full path name of a glossary of the Glossary List.

This procedure has no effect if Name is not the full path name of a glossary of the Glossary List or if Instant Text is not currently running.

Activate_Glossary_Number

```
Pascal Syntax:   procedure Activate_Glossary_Number (Number: Integer);
     C Syntax:   void Activate_Glossary_Number (int Number);
    VB Syntax:   Sub Activate_Glossary_Number (ByVal Number As Integer)
```

The effect of this procedure is to make the glossary specified by the Number active. The effect is similar to that of the shortcut Ctrl+Number where Number must be an integer position in the range 1 to 8.

This procedure has no effect if there is no open glossary associated with the Number position or if Instant Text is not currently running.

Current_Glossary_Shortname

```
Pascal Syntax:   function Current_Glossary_Shortname: Pchar;
     C Syntax:   LPSTR Current_Glossary_Shortname();
```

Returns the short name of the current glossary in the Glossary List. The short name has up to eight letters and does not have an extension.

Returns an empty string if Instant Text is not currently running.

Current_Glossary_Longname

Pascal Syntax: **function** `Current_Glossary_Longname: PChar;`
C Syntax: `LPSTR Current_Glossary_Longname ();`

Returns the full path name of the current glossary in the Glossary List.

Returns an empty string if Instant Text is not currently running.

Current_Glossary_Number

Pascal Syntax: **function** `Current_Glossary_Number: Integer;`
C Syntax: `int Current_Glossary_Number ();`
VB Syntax: `Function Current_Glossary_Number As Integer`

Returns the position of the current glossary in the Glossary List. The position is a number in the range 1 to 15.

Returns the value zero if Instant Text is not currently running.

Save_Active_Glossary_List

Pascal Syntax: **procedure** `Save_Active_Glossary_List;`
C Syntax: `void Save_Active_Glossary_List ();`
VB Syntax: `Sub Save_Active_Glossary_List ()`

The effect of this procedure is to save the current glossary list.

This procedure has no effect if Instant Text is not currently running.

View_Current_Glossary

Pascal Syntax: **procedure** `View_Current_Glossary;`
C Syntax: `void View_Current_Glossary ();`
VB Syntax: `Sub View_Current_Glossary ()`

The effect of this procedure is to open the Glossary Viewer for the current glossary.

This procedure has no effect if Instant Text is not currently running.

Add_Glossary_Entry

Pascal Syntax: **procedure** `Add_Glossary_Entry (AnEntry: PChar);`
C Syntax: `void Add_Glossary_Entry (LPSTR AnEntry);`
VB Syntax: `Sub Add_Glossary_Entry (ByVal AnEntry As String)`

The effect of this procedure is to open the Glossary Viewer and the glossary editor with the string AnEntry as the new entry to be added to the current glossary.

This procedure has no effect if Instant Text is not currently running.

Procedures for Application and Editor Interaction

Several procedures are provided to allow Instant Text to interact with an application.

1. **Link_By_Name** is the linking method to be used when linking Instant Text to one of the supported applications. A specialized version is provided for MS Word: **Link_To_MS_Word**
2. **Link_Instant_Text** is used to communicate to Instant Text the handle of the application that is not a supported application. When Link_Instant_Text is used, the following procedures are also needed:
3. **Set_Insert_By_Message** is used to communicate to Instant Text a message number, which is the message to be issued by Instant Text to insert text. Cancel_Insert_Message cancels the most recent message number.
4. **Get_Expansion_Text** is a function that returns the current expansion text. Get_Expansion_Text will be called by the message handler defined by Set_Insert_By_Message. **Get_Expansion_Text** is a function that returns the length if the current expansion text. **Copy_Expansion_Text** is a procedure that copies to the clipboard the current expansion text.
5. **Set_Backspace_By_Message** is used to communicate to Instant Text a message number, which is the message to be issued by Instant Text to backspace.

Instant Text uses the two messages, the function Get_Expansion_Text, and the procedure Copy_Expansion_Text to send strings to the application.

Link_By_Name

```
Pascal Syntax:    procedure Link_By_Name (Name: PChar);
     C Syntax:    void Link_By_Name (LPSTR Name);
    VB Syntax:    Sub Link_By_Name (ByVal Name As String)
```

The effect of this procedure is to link an application with Instant Text. The application is specified by a string that is either the title of its main window or the start of this title. If several windows match the given name, one of them is linked.

This procedure has no effect if no window matches the given name or if Instant Text is not currently running.

Link_To_MS_Word

```
Pascal Syntax:    procedure Link_To_MS_Word;
     C Syntax:    void Link_To_MS_Word ();
    VB Syntax:    Sub Link_To_MS_Word ()
```

The effect of this procedure is to link Microsoft Word with Instant Text

This procedure has no effect if Microsoft Word or Instant Text is not currently running.

Link_Instant_Text

Pascal Syntax: **function** `Link_Instant_Text (AHandle: HWnd): Boolean;`
C Syntax: `BOOL Link_Instant_Text (HWND AHandle);`
VB Syntax: `Function Link_Instant_Text (ByVal AHandle As Long) As Boolean`

The effect of this function is to link an application with Instant Text and communicate the application handle to Instant Text. The application is specified by its handle in the parameter AHandle.

This function returns false and has no other effect if AHandle is not a valid handle or if Instant Text is not currently running. The function returns true otherwise.

Set_Insert_By_Message

Pascal Syntax: **procedure** `Set_Insert_By_Message (AMessage: Integer);`
C Syntax: `void Set_Insert_By_Message (int AMessage);`
VB Syntax: `Sub Set_Insert_By_Message (ByVal AMessage As Integer)`

The effect of this procedure is to provide Instant Text with a message number to be used for sending text to the application.

This procedure has no effect if Instant Text is not currently running.

Set_Backspace_By_Message

Pascal Syntax: **procedure** `Set_Backspace_By_Message (AMessage: Integer);`
C Syntax: `void Set_Backspace_By_Message (int AMessage);`
VB Syntax: `Sub Set_Backspace_By_Message (ByVal AMessage As Integer)`

The effect of this procedure is to provide Instant Text with a message number to be used for backspacing.

This procedure has no effect if Instant Text is not currently running.

Cancel_Messages

Pascal Syntax: **procedure** `Cancel_Messages;`
C Syntax: `void Cancel_Messages ();`
VB Syntax: `Sub Cancel_Messages ()`

The effect of this procedure is to cancel insertion by messages.

This procedure has no effect if Instant Text is not currently running.

Get_Expansion_Text

```
Pascal Syntax:    function Get_Expansion_Text: PChar;
     C Syntax:    LPSTR Get_Expansion_Text();
```

Returns a string containing the expansion text produced by the last expansion.

This function is intended to be called in the message handler called for text expansion.

Get_Expansion_Length

```
Pascal Syntax:    function Get_Expansion_Length: Integer;
     C Syntax:    int Get_Expansion_Length ();
    VB Syntax:    Function Get_Expansion_Length () As Long
```

Returns the length of the expansion text produced by the last expansion.

This function is intended to be called in the message handler called for text expansion.

Copy_Expansion_Text

```
Pascal Syntax:    procedure Copy_Expansion_Text(AWindow: HWnd; AFormat: UINT);
     C Syntax:    void Copy_Expansion_Text (hWnd AWindow; UINT AFormat);
    VB Syntax:    Sub Copy_Expansion_Text (ByVal hWnd As Long, _
                                          ByVal AFormat As Integer)
```

The effect of this procedure is to copy to the clipboard the expansion text produced by the last expansion.

This procedure is intended to be called in the message handler called for text expansion.

Interaction with MS Word - an example

The short example given below shows how to define macros to start and close Instant Text in Visual Basic for Applications. Then it becomes very simple to include call to these macros in auto macros or to have buttons in a toolbar calling these macros.

The functions and procedures of the API are first declared in Declare statements. (The path name for the dll should be adapted to the current installation.)

```
'-------------------------------------------------------------------
'   DECLARATIONS FOR THE IT API:
'   Change the Library path name as needed:
'-------------------------------------------------------------------

Private Declare Sub Start_Instant_Text Lib "C:\Itxt25.exe\itapidll.dll" ()

Private Declare Sub Restore_Instant_Text Lib "C:\Itxt25.exe\itapidll.dll" ()

Private Declare Sub Link_To_MS_Word Lib "C:\Itxt25.exe\itapidll.dll" ()

Private Declare Function Instant_Text_Running _
            Lib "C:\Itxt25.exe\itapidll.dll" () As Boolean

Private Declare Sub Close_Instant_Text Lib "C:\Itxt25.exe\itapidll.dll" ()
```

StartInstantText either starts or restores Instant Text and then calls Link_To_MS_Word:

```
'-------------------------------------------------------------------
'     Macro: StartInstantText
'-------------------------------------------------------------------
Sub StartInstantText()
    On Error Resume Next

    If Instant_Text_Running Then
        Restore_Instant_Text
    Else
        Start_Instant_Text
    End If

    Link_To_MS_Word
End Sub
```

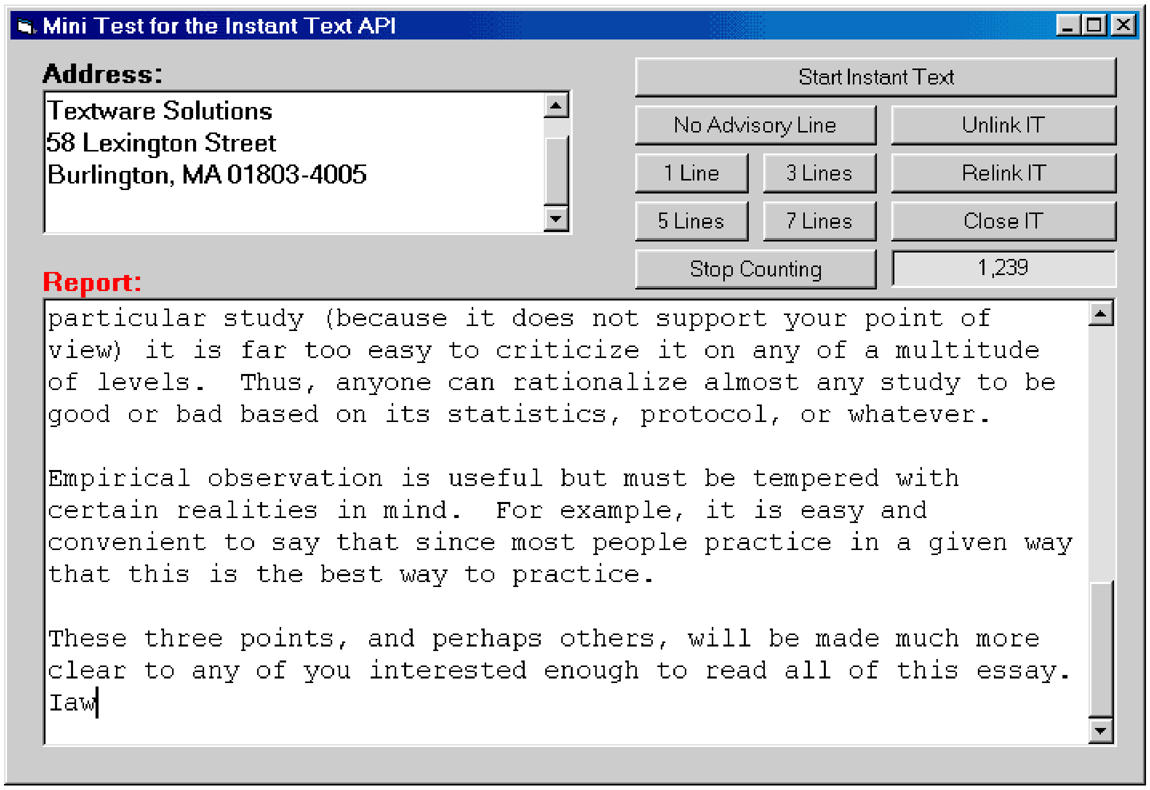
CloseInstantText closes Instant Text if it is currently running:

```
'-------------------------------------------------------------------
'     Macro: CloseInstantText
'-------------------------------------------------------------------

Sub CloseInstantText()
    On Error Resume Next

    If Instant_Text_Running Then
        Close_Instant_Text
    End If
End Sub
```

Mini Test – A Simple Example of Use of the API – Visual Basic Version

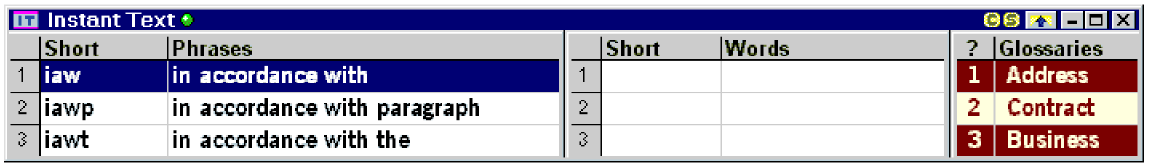
The following example illustrates the use of the API to build a small application that interacts with Instant Text. Mini Test is able to drive Instant Text in several ways:

- Start or close Instant Text with the Start Instant Text and the Close IT buttons. Unlink or relink Mini Test and Instant Text with the Unlink IT and Relink IT buttons.
- As characters are typed in Mini Test, the Instant Text advisories are updated. Here after typing "Iaw" the candidate expansion "In accordance with" is proposed and will replace "Iaw" when the space bar (or some other marker) is typed.
- Pressing No Advisory Line minimizes Instant Text, while still linked. Pressing the buttons "1 Line" through "7 Lines" resizes the Instant Text advisory.
- The current Glossary is set by Mini Test depending on which editor has the focus. The Address glossary is automatically selected when the focus is on the Address text box, the Contract glossary for the Report text box.
- When the Start Counting button is pressed, the number of character produced is diplayed in the gray editor on the right as typing goes along. The caption of the button is changed to Stop Counting as shown below.



Textware Solutions
58 Lexington Street
Burlington, MA 01803-4005

```
particular study (because it does not support your point of
view) it is far too easy to criticize it on any of a multitude
of levels.  Thus, anyone can rationalize almost any study to be
good or bad based on its statistics, protocol, or whatever.

Empirical observation is useful but must be tempered with
certain realities in mind.  For example, it is easy and
convenient to say that since most people practice in a given way
that this is the best way to practice.

These three points, and perhaps others, will be made much more
clear to any of you interested enough to read all of this essay.
Iaw
```



	Short	Phrases
1	iaw	in accordance with
2	iawp	in accordance with paragraph
3	iawt	in accordance with the

	Short	Words
1		
2		
3		

?	Glossaries
1	Address
2	Contract
3	Business

The Visual Basic implementation of Mini Test uses subclassing ot the main window to allow the application to receive user-defined messages. This must be done in a separate module - the code cannot be in the form - and it is done here in a module called Minibase.bas

This module starts with the declaration of the Windows API functions CallWindowProc and SetWindowLong which are needed to implement subclassing. The module also contains the global variables Old_WindowProc, MainWindow_Handle, IT_Minimized, Counting, and Character_Count, and several global constants, in particular, the constant Insert_Message and Backspace_Message which define the messages used to communicate with Instant Text.

```
Attribute VB_Name = "Minibase"

'-------------------------------------------------------------------
'   WINDOWS FUNCTION DECLARATIONS:
'-------------------------------------------------------------------

    Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" _
            (ByVal lpPrevWndFunc As Long, _
             ByVal hWnd As Long, _
             ByVal Msg As Long, _
             ByVal wParam As Long, _
             ByVal lParam As Long) As Long

    Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
            (ByVal hWnd As Long, _
             ByVal nIndex As Long, _
             ByVal dwNewLong As Long) As Long

'-------------------------------------------------------------------
'   WINDOWS CONSTANT DECLARATIONS:
'-------------------------------------------------------------------

    Global Const GWL_WNDPROC = -4

    Global Const CF_TEXT = 1

    Global Const WM_CHAR = &H102
    Global Const VK_BACK = &H8
    Global Const SMTO_BLOCK = &H1
    Global Const SMTO_ABORTIFHUNG = &H2
    Global Const SMTO_ABORTIFHUNG_or_BLOCK = &H3

    Global Old_WindowProc As Long
    Global MainWindow_Handle As Long

    Global Counting As Boolean
    Global IT_Minimized As Boolean

    Global Character_Count As Long

    ' WM_USER is &H400 and VB does not allow adding constants:

    Global Const Insert_Message = &H500         ' WM_USER + &H100
    Global Const Backspace_Message = &H501      ' WM_USER + &H101
```

The code for subclassing then follows:

```
'-------------------------------------------------------------------
'   SUBCLASSING:
'-------------------------------------------------------------------

    Public Sub Start_Subclassing(ByVal WindowHandle As Long)

        MainWindow_Handle = WindowHandle

        Old_WindowProc = SetWindowLong( _
                            MainWindow_Handle, _
                            GWL_WNDPROC, _
                            AddressOf WindowProc)
    End Sub
```

Start_Subclassing establishes WindowProc as the new windows procedure for the application and stores the address of the former windows procedure in the global variable Old_WindowProc

```
    Public Sub End_Subclassing()
        Dim DummyValue As Long

        DummyValue = SetWindowLong( _
                            MainWindow_Handle, _
                            GWL_WNDPROC, _
                            Old_WindowProc)
    End Sub
```

End_Subclassing restores the original windows procedure.

```
    Function WindowProc(ByVal hw As Long, _
                        ByVal uMsg As Long, _
                        ByVal wParam As Long, _
                        ByVal lParam As Long) As Long

        WindowProc = True

        Select Case uMsg
            Case Insert_Message
                Call MiniTest_form.DoInsert(wParam)

            Case Backspace_Message
                Call MiniTest_form.DoBackSpace(wParam, lParam)

            Case Else
                WindowProc = _
                    CallWindowProc( _
                        Old_WindowProc, _
                        hw, _
                        uMsg, _
                        wParam, _
                        lParam)
        End Select
    End Function

'-------------------------------------------------------------------
'   END OF MODULE
'-------------------------------------------------------------------
```

WindowProc processes all messages for the application.

If the message is one of the messages sent by Instant Text the procedures DoInsert or DoBackSpace are called.

Otherwise, the message is passed to the normal windows procedure servicing the application.

The rest of the code is with the form Minitest.frm itself:

```
VERSION 5.00
Begin VB.Form MiniTest_form
```
   ... form coded omitted ...
```
End
...
'------------------------------------------------------------------
'     First, the declarations of the API functions and procedures
'     that are called in the example:
'------------------------------------------------------------------

 Private Declare Function SendMessageTimeout& Lib "user32" Alias "SendMessageTimeoutA" _
            (ByVal hWnd As Long, _
             ByVal Msg As Long, _
             ByVal wParam As Long, _
             ByVal lParam As Long, _
             ByVal fuFlags As Long, _
             ByVal uTimeout As Long, _
             ByVal lpdwResult As Long)

'------------------------------------------------------------------
'   STARTING, LINKING, and CLOSING:
'------------------------------------------------------------------

Private Declare Function Instant_Text_Running _
                Lib "C:\InstText\Exe32\itapidll.dll" () As Boolean

Private Declare Sub Start_Instant_Text_Lower _
                Lib "C:\InstText\Exe32\itapidll.dll" (ByVal Lines As Integer)

Private Declare Function Link_Instant_Text _
                Lib "C:\InstText\Exe32\itapidll.dll" (ByVal AHandle As Long) As Boolean

Private Declare Sub Unlink_Instant_Text _
                Lib "C:\InstText\Exe32\itapidll.dll" ()

Private Declare Sub Close_Instant_Text _
                Lib "C:\InstText\Exe32\itapidll.dll" ()

'------------------------------------------------------------------
'   MINIMIZE and RESTORE:
'------------------------------------------------------------------

Private Declare Sub Minimize_Instant_Text _
                Lib "C:\InstText\Exe32\itapidll.dll" ()

Private Declare Sub Restore_Instant_Text _
                Lib "C:\InstText\Exe32\itapidll.dll" ()

'------------------------------------------------------------------
'   ADVISORY LINES:
'------------------------------------------------------------------

Private Declare Sub Set_Advisory_Lines _
                Lib "C:\InstText\Exe32\itapidll.dll" (ByVal Lines As Integer)

'------------------------------------------------------------------
'   GLOSSARY HANDLING:
'------------------------------------------------------------------

Private Declare Sub Activate_Glossary_Number _
                Lib "C:\InstText\Exe32\itapidll.dll" (ByVal Number As Integer)
```

```
'------------------------------------------------------------------
'   FOR APPLICATION AND EDITOR INTERACTION:
'------------------------------------------------------------------

Private Declare Sub Set_Insert_By_Message _
                Lib "C:\InstText\Exe32\itapidll.dll" (ByVal AMessage As Integer)

Private Declare Sub Set_Backspace_By_Message _
                Lib "C:\InstText\Exe32\itapidll.dll" (ByVal AMessage As Integer)

Private Declare Function Get_Expansion_Length _
                Lib "C:\InstText\Exe32\itapidll.dll" () As Long

Private Declare Sub Copy_Expansion_Text _
                Lib "C:\InstText\Exe32\itapidll.dll" _
                   (ByVal AWindow As Long, _
                    ByVal AFormat As Integer)


'------------------------------------------------------------------
'  UPON FORM LOAD AND UNLOAD:
'------------------------------------------------------------------

Private Sub Form_Load()
    Call Start_Subclassing(hWnd)

    Counting = False
    Character_Count = 0
    Count_editor.Text = FormatNumber(Character_Count, 0)
End Sub
```

When the form is loaded, subclassing is established and other initializations are done.

```
Private Sub Form_Unload(Cancel As Integer)
   Call End_Subclassing
End Sub
```

Subclassing must be undone at the end to allow proper termination.

```
'------------------------------------------------------------------
'  THE INSERTION PROCEDURES:
'------------------------------------------------------------------

Public Sub DoInsert(ByVal wParam As Long)

    Call Copy_Expansion_Text(hWnd, CF_TEXT)


    If Counting Then
        Character_Count = Character_Count + Get_Expansion_Length()
        Count_editor.Text = FormatNumber(Character_Count, 0)
    End If


    If wParam = Address_text.hWnd Then
        Address_text.SelText = Clipboard.GetText()
    Else
        Report_text.SelText = Clipboard.GetText()
    End If
End Sub
```

This is the procedure called by message whenever Instant Text expands text. The first API call copies the text to the clipboard. The second API call, to Get_Expansion_Length, is used to update the character count.

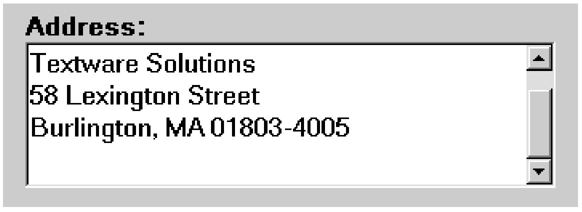
Finally, the text is pasted into the proper editor.

```
Public Sub DoBackSpace(ByVal wParam As Long, ByVal lParam As Long)

    Dim Index As Integer
    Dim Backspaces As Long
    Dim Status As Long
    Dim DummyValue As Long

    Backspaces = lParam
    For Index = 1 To Backspaces

        DummyValue = SendMessageTimeout( _
                        wParam, _
                        WM_CHAR, _
                        VK_BACK, _
                        lParam, _
                        SMTO_ABORTIFHUNG_or_BLOCK, _
                        1000, _
                        Status)
    Next Index
End Sub
```

This is the procedure called by message by Instant Text to backspace the short code before doing the expansion.

Note the use of a message with a timeout of one second since this message goes from Mini Test to a different application (Instant Text).

```
'-------------------------------------------------------------------
'  WHEN EDITORS GET OR LOOSE FOCUS:
'-------------------------------------------------------------------
```

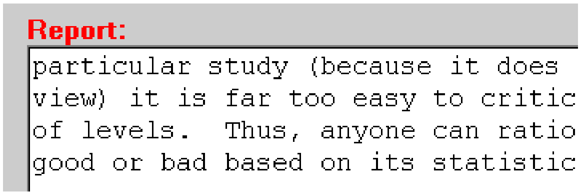


```
Private Sub Address_text_GotFocus()
    Address_label.ForeColor = RGB(255, 0, 0)
    If Instant_Text_Running Then
       Activate_Glossary_Number (1)
    End If
End Sub

Private Sub Address_text_LostFocus()
    Address_label.ForeColor = RGB(0, 0, 0)
End Sub
```

When the editor gets the focus:
(1) The Address label is painted red
(2) The glossary Address is activated and is used while typing in the editor.

When losing focus, the label is colored black again

Similar actions are done for the Report text box.


Report:
particular study (because it does
view) it is far too easy to critic
of levels. Thus, anyone can ratio
good or bad based on its statistic

```
Private Sub Report_text_GotFocus()
    Report_label.ForeColor = RGB(255, 0, 0)
    If Instant_Text_Running Then
       Activate_Glossary_Number (2)
    End If
End Sub

Private Sub Report_text_LostFocus()
    Report_label.ForeColor = RGB(0, 0, 0)
End Sub
```

```
'-----------------------------------------------------------------
'  FINALLY, FOR EACH BUTTON:
'-----------------------------------------------------------------
```

```
Private Sub Start_IT_button_Click()
    If Instant_Text_Running Then
        Restore_Instant_Text
    Else
        Start_Instant_Text_Lower (3)
        IT_Minimized = False
    End If

    Link_Instant_Text (hWnd)

    Set_Insert_By_Message (Insert_Message)
    Set_Backspace_By_Message (Backspace_Message)

End Sub
```

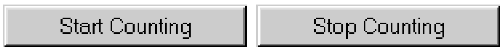
If Instant Text is already running we restore it. Otherwise we start it with 3 advisory lines.

Then we link Mini Test and Instant Text and establish the messages numbers to be used when communicating.

```
Private Sub Close_button_Click()
    If Instant_Text_Running Then
        Close_Instant_Text
    End If
End Sub
```

```
Private Sub Unlink_button_Click()
    If Instant_Text_Running Then
        Unlink_Instant_Text
    End If
End Sub
```

```
Private Sub Relink_button_Click()
    If Instant_Text_Running Then
        Link_Instant_Text (hWnd)
    End If
End Sub
```

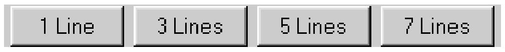


```
Private Sub Count_button_Click()
    If Counting Then
        Counting = False
        Count_button.Caption = "Start Counting"
    Else
        Counting = True
        Count_button.Caption = "Stop Counting"
        Character_Count = 0
        Count_editor.Text = FormatNumber(Character_Count, 0)
    End If
End Sub
```

This implement the "toggle" effect. The caption is either Start or Stop.

```
Private Sub NoLine_button_Click(Index As Integer)
    If Instant_Text_Running Then
        Minimize_Instant_Text
        IT_Minimized = True
    End If
End Sub
```

When No Advisory Line is pressed this makes an API call to minimize Instant Text but does not unlink. So text expansion continues.



```
Private Sub Line_1_button_Click()
    If Instant_Text_Running Then
        Set_Advisory_Lines (1)
        If IT_Minimized Then
            IT_Minimized = False
            Restore_Instant_Text
        End If
    End If
End Sub

Private Sub Line_3_button_Click()
    If Instant_Text_Running Then
        Set_Advisory_Lines (3)
        If IT_Minimized Then
            IT_Minimized = False
            Restore_Instant_Text
        End If
    End If
End Sub

Private Sub Line_5_button_Click()
    If Instant_Text_Running Then
        Set_Advisory_Lines (5)
        If IT_Minimized Then
            IT_Minimized = False
            Restore_Instant_Text
        End If
    End If
End Sub

Private Sub Line_7_button_Click()
    If Instant_Text_Running Then
        Set_Advisory_Lines (7)
        If IT_Minimized Then
            IT_Minimized = False
            Restore_Instant_Text
        End If
    End If
End Sub
```

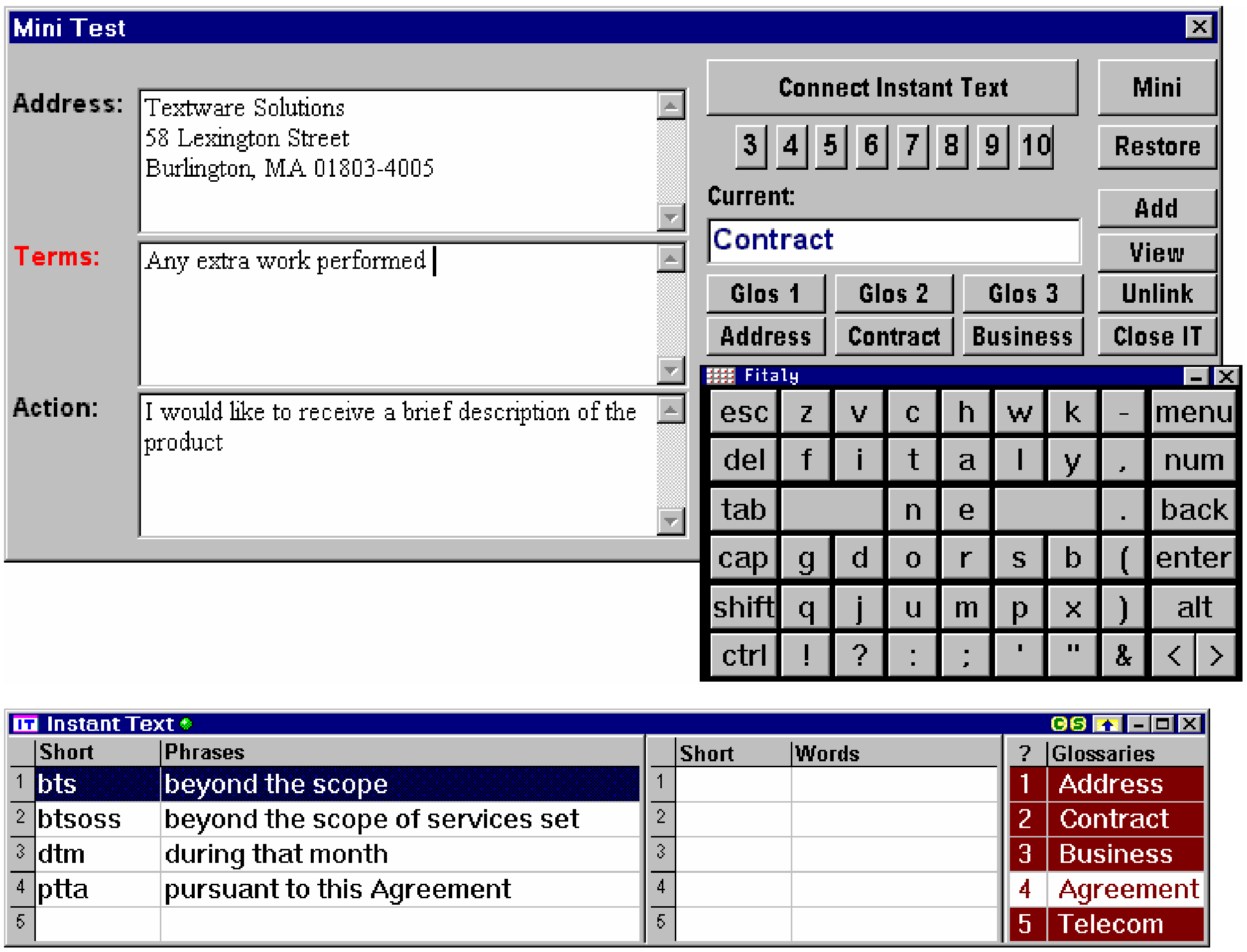
Each of these methods makes an API call to set the proper number of advisory lines.

In addition, if Instant Text is minimized, a second API call is made to restore it.

Mini Test – A Simple Example of Use of the API – Delphi Version

The following example illustrates the use of the API to build a small Pen-based application that interacts with Instant Text and the Fitaly keyboard. The Fitaly keyboard is overlapped on the Mini Test application and is used to enter characters. Mini Test is able to drive Instant Text in several ways:

- The Connect Instant Text button starts Instant Text. The Mini and Restore buttons can be used to minimize or restore it. Top and Non Top decide whether or not Instant Text is a topmost application.
- The buttons labeled 3 to 10 are used to set the number of advisory lines
- The View button opens the Glossary Viewer and Add opens the glossary editor.
- The current Glossary can be selected by the buttons F2, F3, and F4, as well as the buttons Address, Contract, and Business..
- Tapping in one of the memos also establishes a corresponding glossary and requests that all input from Instant Text be directed to the memo. In the example below, the focus is on Terms and this establishes Contract as the current glossary. The Phrase Advisory shows possible continuations for the phrase "extra work performed". Any of these continuations can be selected with a single tap on the corresponding line.



Address: Textware Solutions
58 Lexington Street
Burlington, MA 01803-4005

Terms: Any extra work performed

Action: I would like to receive a brief description of the product

Current: Contract

	Short	Phrases
1	bts	beyond the scope
2	btsoss	beyond the scope of services set
3	dtm	during that month
4	ptta	pursuant to this Agreement
5		

	Short	Words
1		
2		
3		
4		
5		

?	Glossaries
1	Address
2	Contract
3	Business
4	Agreement
5	Telecom

Mini Test in Delphi

The example given below is programmed as a Delphi application. It includes calls to the functions and procedures of the IT_API dll. To simplify recognition of such calls they are indicated with the full dot notation. For example, IT_API.Restore_Instant_Text.

```
unit Testapi;

interface
usesSysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
        Forms, Dialogs, StdCtrls, IT_API;

const Insert_Message     = WM_User + $100;
const Backspace_Message  = WM_User + $101;

type TAPI_Test = class(TForm)

   Address_label     : TLabel;     Terms_label : TLabel;     Action_label   : TLabel;
   Address_Edit      : TMemo;      Terms_Edit  : TMemo;      Action_Edit    : TMemo;

   Address           : TButton;    Contract    : TButton;    Business       : TButton;
   Start_IT          : TButton;    Glos1, Glos2, Glos3  : TButton;

   Mini              : TButton;    Restore     : TButton;
   View_Button       : TButton;    Add_Button  : TButton;

   Size3             : TButton;    Size4       : TButton;
   Size5             : TButton;    Size6       : TButton;
   Size7             : TButton;    Size8       : TButton;
   Size9             : TButton;    Size10      : TButton;

   Current_Edit      : TEdit;

   procedure FormCreate      (Sender: TObject);     procedure FormDestroy(Sender: TObject);

   procedure Start_IT_click     (Sender: TObject);
   procedure View_Button_click  (Sender: TObject);
   procedure Add_Button_click   (Sender: TObject);
   procedure Address_Edit_click (Sender: TObject);
   procedure Terms_Edit_click   (Sender: TObject);
   procedure Action_Edit_click  (Sender: TObject);

   procedure Address_click      (Sender: TObject);     procedure Glos_1_Click(Sender: TObject);
   procedure Contract_click     (Sender: TObject);     procedure Glos_2_Click(Sender: TObject);
   procedure Business_click     (Sender: TObject);     procedure Glos_3_Click(Sender: TObject);

   procedure Mini_click      (Sender: TObject);     procedure Restore_click(Sender: TObject);

   procedure Size3_click        (Sender: TObject);     procedure Size4_click  (Sender: TObject);
   procedure Size5_click        (Sender: TObject);     procedure Size6_click  (Sender: TObject);
   procedure Size7_click        (Sender: TObject);     procedure Size8_click  (Sender: TObject);
   procedure Size9_click        (Sender: TObject);     procedure Size10_click (Sender: TObject);

private
   ITAPI_library_handle : THandle;

   procedure Link_to_Address;
   procedure Link_to_Terms;
   procedure Link_to_Action;

   procedure Connect_to_IT;

   function Client_Editor_of (AHandle: THandle): TMemo;
public
   procedure DoInsert     (var Msg: TMessage); message Insert_Message;
   procedure DoBackSpace  (var Msg: TMessage); message Backspace_Message;
end;

var  API_Test      : TAPI_Test;
```

```
implementation
   uses Clipbrd;

{$R *.DFM}


function  TAPI_Test.Client_Editor_of (AHandle: THandle): TMemo;
begin
   if       AHandle =  Terms_Edit.Handle     then
                       Client_Editor_of := Terms_Edit
   else if AHandle =  Address_Edit.Handle   then
                       Client_Editor_of := Address_Edit
   else
                       Client_Editor_of := Action_Edit;
end;
```

// The Insertion Procedure:

```
procedure TAPI_Test.DoInsert (var Msg: TMessage);
   var AString        : PChar;
   var Client         : TMemo;
begin
   AString := IT_API.Get_Expansion_Text;
   CLIPBRD.Clipboard.SetTextBuf(AString);

   Client := Client_Editor_of (THandle(Msg.WParam));
   Client.PasteFromClipboard;
end;
```

This is the procedure that will be called by messages issued by Instant Text for insertions.

The insertion procedure calls Client_Editor_of to find out which editor will receive the insertion.

The insertion itself is done here using the clipboard.

// The Backspace Procedure:

```
procedure TAPI_Test.DoBackSpace (var Msg: TMessage);
   var   Index          : Integer;
   var   Backspaces     : Integer;
   var   Status          : DWord;
begin
  Backspaces := Integer(Msg.LParam);
  for Index := 1 to Backspaces do
      SendMessageTimeout(
         THandle(Msg.WParam),              // Window
         WM_Char,                          // Message
         WPARAM(VK_Back),                  // WParam
         LPARAM(1),                        // LParam
         SMTO_ABORTIFHUNG or SMTO_BLOCK,   // Flags
         1000,                             // Timeout
         Status);                          // Result
end;
```

This is the procedure that will be called by messages issued by Instant Text for backspaces.

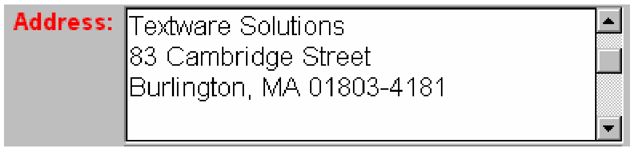
Note the use of a message with time-out since this goes to another application.

// Create and Destroy:

```
procedure TAPI_Test.FormCreate(Sender: TObject);
begin
   ITAPI_library_handle := LoadLibrary('itapidll.dll');
   if ITAPI_library_handle < 32 then
         ShowMessage('Loading Instant Text API Failed.');

   Address_Edit.  ScrollBars := ssVertical;
   Terms_Edit.    ScrollBars := ssVertical;
   Action_Edit.   ScrollBars := ssVertical;
   Height   := Action_Edit.Height + Terms_Edit.Height
               + Address_Edit.Height + 2*Start_IT.Height;
   Width := Address_label.Width + Address_Edit.Width +
            ((9*View_Button.Width) div 2) ;
end;

procedure TAPI_Test.FormDestroy(Sender: TObject);
begin
   FreeLibrary(ITAPI_library_handle);
end;
```

// The Link_to_... Methods:

```
procedure TAPI_Test.Connect_to_IT;
begin
   if IT_API.Instant_Text_Running then
      IT_API.Restore_Instant_Text;
   else
      IT_API.Start_Instant_Text;

   IT_API.Link_Instant_Text(Handle);

   IT_API.Set_Insert_by_Message(Insert_Message);
   IT_API.Set_Backspace_by_Message(Backspace_Message);
end;
```

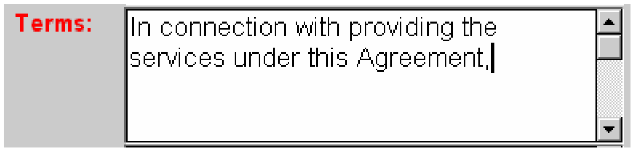


Textware Solutions
83 Cambridge Street
Burlington, MA 01803-4181

When the pen taps on the address memo:
(1) The Address label is painted red
(2) The glossary Address is activated and this is reflected in the Current Editor
(3) The address memo is established as the client editor for Instant Text

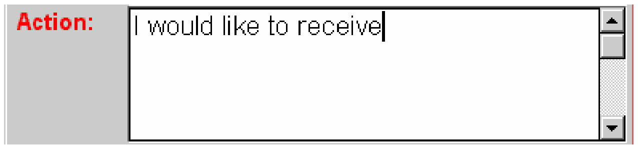
Similar actions are done for the Terms memo and the Action memo, with the corresponding glossaries: Contract for Terms and Business for Action.

```
procedure TAPI_Test.Link_to_Address;
begin
   Address_label.   Font.Color  := clRed;
   Terms_label.     Font.Color  := clBlack;
   Action_label.    Font.Color  := clBlack;

   IT_API.Activate_Glossary_ShortName('Address');
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;

   IT_API.Set_Start_State;
   Address_Edit.SetFocus;
end;
```



In connection with providing the services under this Agreement,

```
procedure TAPI_Test.Link_to_Terms;
begin
   Address_label.   Font.Color := clBlack;
   Terms_label.     Font.Color := clRed;
   Action_label.    Font.Color := clBlack;

   IT_API.Activate_Glossary_ShortName('Contract');
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;

   IT_API.Set_Start_State;
   Terms_Edit.SetFocus;
end;
```



```
procedure TAPI_Test.Link_to_Action;
begin
   Address_label.    Font.Color := clBlack;
   Terms_label.      Font.Color := clBlack;
   Action_label.     Font.Color := clRed;

   IT_API.Activate_Glossary_ShortName('Business');
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;

   IT_API.Set_Start_State;
   Action_Edit.SetFocus;
end;
```

// The Click methods:

```
procedure TAPI_Test.Start_IT_click(Sender: TObject);
begin
   Connect_to_IT;  Link_to_Address;
end;
```

Establishing the current glossary by means of the procedure Activate_Glossary_Key

Then reflecting the change in the current glossary editor

```
procedure TAPI_Test.Glos_1_click(Sender: TObject);
begin
   IT_API.Activate_Glossary_Number(1);
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;
end;

procedure TAPI_Test.Glos_2_click(Sender: TObject);
begin
   IT_API.Activate_Glossary_Number(2);
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;
end;

procedure TAPI_Test.Glos_3_click(Sender: TObject);
begin
   IT_API.Activate_Glossary_Number(3);
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;
end;
```

Establishing the current glossary by means of the procedures Activate_Glossary_ShortName and Activate_Glossary_LongName.

Then reflecting the change in the current glossary editor.

```
procedure TAPI_Test.Address_click(Sender: TObject);
begin
   IT_API.Activate_Glossary_Shortname('Address');
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;
end;

procedure TAPI_Test.Contract_click(Sender: TObject);
begin
   IT_API.Activate_Glossary_Shortname('Contract');
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;
end;
```

```
procedure TAPI_Test.Business_click(Sender: TObject);
begin
   IT_API.Activate_Glossary_Longname('c:\insttext\glossary\business.glo');
   Current_Edit.Text := IT_API.Current_Glossary_Shortname;
end;
```

// The Click method for the three memos:

Clicking in any memo connects to Instant Text (if not already started) and links the corresponding memo.

```
procedure TAPI_Test.Address_Edit_click(Sender: TObject);
begin
   Connect_to_IT;  Link_to_Address;
end;

procedure TAPI_Test.Terms_Edit_click(Sender: TObject);
begin
   Connect_to_IT;  Link_to_Terms;
end;

procedure TAPI_Test.Action_Edit_click(Sender: TObject);
begin
    Connect_to_IT;  Link_to_Action;
end;
```

Mini

```
procedure TAPI_Test.Mini_click(Sender: TObject);
begin
   IT_API.Minimize_Instant_Text;
end;
```

Restore

```
procedure TAPI_Test.Restore_click(Sender: TObject);
begin
   IT_API.Restore_Instant_Text;
end;
```

3 4 5 6 7 8 9 10

Each of these methods establishes a corresponding number of lines for the Instant text advisories.

```
procedure TAPI_Test.Size3_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(3);
end;

procedure TAPI_Test.Size4_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(4);
end;

procedure TAPI_Test.Size5_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(5);
end;

procedure TAPI_Test.Size6_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(6);
end;

procedure TAPI_Test.Size7_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(7);
end;
```

```
procedure TAPI_Test.Size8_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(8);
end;

procedure TAPI_Test.Size9_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(9);
end;

procedure TAPI_Test.Size10_click(Sender: TObject);
begin
   IT_API.Set_Advisory_Lines(10);
end;
```

View

```
procedure TAPI_Test.View_Button_click(Sender: TObject);
begin
   IT_API.View_Current_Glossary;
end;
```

Add

```
procedure TAPI_Test.Add_Button_click (Sender: TObject);
   var buffer : array [0..200] of Char;
begin
   StrCopy(buffer,'Here is a new glossary entry');
   IT_API.Add_Glossary_Entry(buffer);
end;
```

This method calls Add_Glossary_Entry with a standard message, but your application can use a similar technique to insert a context-dependent string.

Unlink

```
procedure TAPI_Test.Unlink_Button_click (Sender: TObject);
begin
   IT_API.Unlink_Instant_Text;
end;
```

Close IT

```
procedure TAPI_Test.CloseIT_Button_click    (Sender: TObject);
begin
   IT_API.Close_Instant_Text;
end;


end.
```

Visual Basic Listing of the API for Instant Text

The functions and procedures available in Visual Basic are listed below. A few functions that return a string have been left out..

```
'------------------------------------------------------------------
'    DECLARATIONS FOR THE IT API:
'    The Library path name is indicated here as "IT API dll"
'    and should be changed into the actual path.
'------------------------------------------------------------------

Declare Function Instant_Text_Running       Lib "IT API dll" () As Boolean
Declare Sub Start_Instant_Text              Lib "IT API dll" ()
Declare Sub Start_Instant_Text_Lower        Lib "IT API dll" (ByVal Lines As Integer)
Declare Function Link_Instant_Text          Lib "IT API dll" (ByVal AHandle As Long) As Boolean
Declare Sub Link_By_Name                    Lib "IT API dll" (ByVal Name As String)
Declare Sub Link_To_MS_Word                 Lib "IT API dll" ()

Declare Sub Unlink_Instant_Text             Lib "IT API dll" ()

Declare Sub Minimize_Instant_Text           Lib "IT API dll" ()
Declare Sub Restore_Instant_Text            Lib "IT API dll" ()
Declare Sub Set_IT_Lower_Layout             Lib "IT API dll" ()
Declare Sub Set_IT_Full_Screen              Lib "IT API dll" ()
Declare Sub Instant_Text_Top                Lib "IT API dll" ()
Declare Sub Instant_Text_Nontop             Lib "IT API dll" ()

Declare Function Instant_Text_Handle        Lib "IT API dll" () As Long

Declare Function Get_Advisory_Lines         Lib "IT API dll" () As Integer
Declare Sub Set_Advisory_Lines              Lib "IT API dll" (ByVal Lines As Integer)

Declare Sub Set_Start_State                 Lib "IT API dll" ()
Declare Sub Close_Instant_Text              Lib "IT API dll" ()
'-----------------------------------------------------------------------------
'   GLOSSARY HANDLING:
'-----------------------------------------------------------------------------
Declare Sub Open_Named_Glossary             Lib "IT API dll" (ByVal Name As String)
Declare Sub View_Current_Glossary           Lib "IT API dll" ()
Declare Sub Activate_Glossary_Shortname     Lib "IT API dll" (ByVal Name As String)
Declare Sub Activate_Glossary_Longname      Lib "IT API dll" (ByVal Name As String)
Declare Sub Activate_Glossary_Number        Lib "IT API dll" (ByVal Number As Integer)

Declare Function Current_Glossary_Number    Lib "IT API dll" () As Integer

Declare Sub Save_Active_Glossary_List       Lib "IT API dll" ()

' Adding to the Current Glossary:

Declare Sub Add_Glossary_Entry              Lib "IT API dll" (ByVal AnEntry As String)
' Closing and reopening the Current Glossary:

Declare Sub Close_Current_Glossary          Lib "IT API dll" ()
Declare Sub Reopen_Current_Glossary         Lib "IT API dll" ()

' For Application and Editor Interaction:

Declare Sub Set_Insert_By_Message           Lib "IT API dll" (ByVal AMessage As Integer)
Declare Sub Cancel_Messages                 Lib "IT API dll" ()
Declare Sub Set_Backspace_By_Message        Lib "IT API dll" (ByVal AMessage As Integer)

Declare Function Get_Expansion_Length       Lib "IT API dll" () As Integer

Declare Sub Copy_Expansion_Text             Lib "IT API dll"  (ByVal AWindow As Long, _
                                                               ByVal AFormat As Integer)
```

Pascal Listing of the API

```
unit IT_API;
interface
uses WinTypes;

// Starting and Configuring:

function   Instant_Text_Running              : Boolean;
procedure  Start_Instant_Text;
procedure  Start_Instant_Text_Lower          (Lines        : Integer);
function   Link_Instant_Text                 (AHandle      : Integer): Boolean;
procedure  Link_By_Name                      (Name         : PChar);
procedure  Link_To_MS_Word;
procedure  Set_Start_State;

procedure  Unlink_Instant_Text;
procedure  Close_Instant_Text;

procedure  Minimize_Instant_Text;
procedure  Restore_Instant_Text;
procedure  Set_IT_Lower_Layout;
procedure  Set_IT_Full_Screen;

function Instant_Text_Handle                 : HWnd;
procedure  Get_IT_Rectangle                  (var Rect   : TRect);

function   Get_Advisory_Lines                : Integer;
procedure  Set_Advisory_Lines                (Lines        : Integer);

// Glossary Handling:

procedure  Open_Named_Glossary               (Name         : PChar);
procedure  View_Current_Glossary;
procedure  Activate_Glossary_Shortname       (Name         : PChar);
procedure  Activate_Glossary_Longname        (Name         : PChar);
procedure  Activate_Glossary_Number          (Number       : Integer);
procedure  Save_Active_Glossary_List;

function Current_Glossary_Shortname          : PChar;
function Current_Glossary_Longname           : PChar;
function Current_Glossary_Number             : Integer;

// Adding to the Current Glossary:

procedure  Add_Glossary_Entry       (AnEntry    : PChar);

// Closing and Reopening the Current Glossary:

procedure  Close_Current_Glossary;
procedure  Reopen_Current_Glossary;

// For Application and Editor Interaction:

procedure  Set_Insert_By_Message         (AMessage   : Word);
procedure  Cancel_Messages;
procedure  Set_Backspace_By_Message      (AMessage   : Word);

function   Get_Expansion_Text            : PChar;
procedure  Copy_Expansion_Text           (AWindow    : HWnd; AFormat: UINT);
function   Get_Expansion_Length          : Integer;

...
end.
```