Registration

Registering by Credit Card

You can register your copy of TVisAwk either by check, credit card or through compuserve (if you are a member). The registration fee is $ 40 (US) and includes full source upon registration. To register through compuserve GO SWREG and use #10931 to register on line. Compuserve will notify me of your registration (usually within minutes) and I will filemail the full distribution including the source code.

If you want to register by sending a check be aware that I will not send the full distribution to you until your check is received. Send checks to:

John C. Taylor
3475 Holcomb Br Rd
Suite 202
Norcross, GA USA 30092

Again, if you need a diskette mailed to you please include $ 5 (US) for postage and handling.

TVisAwk Component

Properties Methods Events

Unit
PsVisAwk

Description:

TVisAwk is a component that can be used for parsing text or text files. There is also a Like method that adds pattern matching abilities similar to the 'Like' operator in Visual Basic. With TVisAwk you not only eliminate tedious parsing code but you can give the application user visual feedback for lengthy file parsing jobs. The visual display also makes debugging a snap since you can see the actions of the parser at run time instead of having to resort to the debug and watch windows in Delphi.

Copyright and Licensing:

TVisAwk is copyrighted material and is distributed as shareware. You are free to distribute TVisAwk to your friends as long as you distribute it in the original zip file containing this help file.

TVisAwk Properties

Action
ConvertCase
EndLine
ErrorNumber
ErrorText
F
Filename
FilterQuotes
FS
KeepVisuals
MatchQuotes
LineNumber
N
OutputEmbraceChars
OutputFile
OutputOverwrite
OutputSeparator
Pattern
Progress
QuoteChars
StartLine
StepMode
Visuals
VisualCaption
VisualLeft
VisualTop

F Property

Description:

The F property (runtime only) represents the text to be parsed by TAwk. If you set this property directly do as follows:

StrPCopy(VisAwk1.F,'the string to be parsed');

Note that the property is a PChar. If you are parsing a file, you do not need to manipulate this property as the component sets the property equal to the next line in the file to be parsed.

See Also:

Action Property

Action Property

Description:

The action property determines what type of action TVisAwk is to take, valid values are:

0 - Parse text
1 - Scan and Parse file
2 - Stop scanning and close file
3 - Scan file for Pattern

Note: Setting the action property does not cause the indicated function to occur, you must call the execute method to actually perform the desired action.

See Also:

Execute method

Execute method

Description:

The execute method causes TVisAwk to perform the action specified in the Action property.

FS Property

Description:

The FS Property determines which characters TVisAwk will consider to be field separators. The default is spaces and tabs. You can set this string property to be anything you want.

Example:

```
VisAwk1.FS := #32+#9+'()';
```

The example sets space, tab and parentheses as field separators. A field separator will cause the component to split adjacent words into separate tokens. For example: Assuming the field separator is the default (spaces and tab characters) the following string:

'Delphi is a great language.'

will be parsed into 5 tokens...

Delphi
is
a
great
language.

If you don't want the trailing period as part of the token then include it in the FS property.

N Property

Description:

N is an integer property (runtime and read only) which contains the number of tokens found in a line of text which has been parsed by TVisAwk.

Example:

```
StrPCopy(VisAwk1.F,'a line to be parsed');
VisAwk1.FS := ' ';
VisAwk1.Action := 0;     (* parse text *)
VisAwk1.Execute;           (* perform the action *)
for i := 0 to VisAwk1.N - 1 do   (* add to list box *)
   listbox1.items.add(VisAwk1.Gettoken(i));
```

The above code parses the line of text and adds each token to a list box.

See Also:

GetToken Method

GetToken Method

Description:

You can use the gettoken method to retrieve a list of the tokens that were parsed from a particular line of text or from a line in a text file. This typically is done in the OnScan event handler which is fired after TVisAwk reads and parses a line from a file.

Example:

```
(* This code might be in a button click handler*)

VisAwk1.filename := 'readme.txt';
VisAwk1.FS := ' ';     (* space *)
VisAwk1.Action := 2;   (*parse file *)
VisAwk1.Execute;

procedure VisAwk1.OnScan;
var
  i : integer;
begin
  for i := 0 to VisAwk1.N -1 do
    listbox1.items.add(VisAwk1.Gettoken(i));
end;
```

This example catches the tokens after each line of input is parsed from a file.

See Also:

N Property
GetItemPos Method

Progress Property

Description:

Progress is a longint property that reports the percent of the file which has been scanned. This property can be monitored in the OnScan event to report the progress of the current file.

See Also:

OnScan Event

FilterQuotes Property

Description:

FilterQuotes is a boolean property. If FilterQuotes is true, TVisAwk will strip single or double quotes from parsed tokens. Setting this property to false has no effect when MatchQuotes is false.

Example:

When MatchQuotes is true and FilterQuotes is true, a string such as:

"this is a Delphi Component"

would return

this is a Delphi Component

as **one token** without the quotes. If in the example FilterQuotes is false, the resulting token would contain the quotes. Note that when MatchQuotes is true, everything inside the quotes is considered one token regardless of the setting of the FS property. The setting of the FilterQuotes property is ignored when MatchQuotes is false.

See Also:

MatchQuotes property
FS property

MatchQuotes Property

Description:

MatchQuotes is a boolean property. When true, TVisAwk considers everything within matching quotes as a single token. This applies to single and double quotes. You can alter this behavior by setting the QuoteChars property.

See Also:

QuoteChars Property

QuoteChars Property

Description:

QuoteChars is a string which represents the characters to be used by TVisAwk when matching quotes. The default is the double quote character (#34). When the MatchQuotes property is false the setting of this property is ignored. You should use care when changing this property. Consider the following examples:

When QuoteChars is " the string { "O'neal is a Delphi programmer" } would
be parsed into one token:

O'neal is a Delphi programmer

However, if QuoteChars is set to ' and " (#39+#34) the same string would be parsed into six tokens:

O
neal
is
a
Delphi
programmer

If QuoteChars is set to ' (char #39) the string would be parsed into two tokens:
O
neal is a Delphi programmer

Note:

If TVisAwk encounters the end of the line of text without finding a matching end quote character, the entire string after the first quote character will be included in the token as if there was in fact a matching end of quote.
For example, the string:

"this is a string.

would be parsed into one token -

this is a string.

See Also:

MatchQuotes Property
FilterQuotes Property

Filename Property

Description:

The filename property is used in conjunction with an action property setting of 2 or 3. It is the fully qualified pathname and filename of the file to be scanned.

EndLine Property

Description:

The EndLine property is a long integer and tells TVisAwk at what line in the file to end its scan. If Endline is zero TVisAwk scans the entire file until an end of file marker is found.

Example:

```
VisAwk1.Filename := 'c:\readme.txt';
VisAwk1.Action := 1;
VisAwk1.ConvertCase := ccUpper;
VisAwk1.EndLine := 5;
VisAwk1.Execute;
```

This code causes VisAwk to scan a file from the beginning to line 5, converting to upper case before parsing.

See Also:

StartLine Property

ErrorNumber Property

Description:

ErrorNumber is an integer property that is runtime and read only and contains the number of the last error encountered by TVisAwk.

The possible values are:

1 - The filename property specifies a non-existent file.
2 - The StartLine property specifies an invalid line number
3 - File I/O Error

See Also:

OnError Event

ErrorText Property

Description:

ErrorText is a string property containing a description of the last error encountered by TVisAwk. It is runtime and read only.

See Also:

ErrorNumber Property
OnError Event

ConvertCase Property

Description:

The ConvertCase property determines whether VisAwk converts text to upper or lower case before parsing. The valid values are:

ccUpper - convert to upper case
ccLower - convert to lower case
ccNone - do not do case conversion

Note: You can also use this property when doing pattern matching with the Like method. If you want the pattern matching to be case insensitive set this property to either ccUpper or ccLower (it doesn't matter which). If you want case sensitivity set this property to ccNone.

KeepVisuals Property

Description:

The KeepVisuals property is a boolean value that determines whether the visual display of the parsing activity is to be kept on screen when the activity is finished. For example, you might want your application to allow the user to pause the parsing of a file and use step mode to step through lines in a file to view parsing or pattern matching activity.

When KeepVisuals is false and the Visual property is true the visual display form is shown but the Done, Pause, Resume, Step and Replay buttons are not shown. When the parsing activity is complete, the visual display is removed. When KeepVisuals is true and the Visual property is true, the user of your application will have to click the Done button to remove the display.

If the Visual property is false, the value of the KeepVisuals property has no effect.

See Also:

Visuals Property

Pattern Property

Description:

The pattern property is a string that specifies a pattern of text and/or wildcard characters for use with the Like method.

Pattern matching provides a useful tool for string comparisons.Pattern-matching features allow you to use wildcard characters, such as those recognized by DOS. The following table shows the wildcard characters and what they match.

?	Any Single Character
*	Zero or more characters
#	Any single digit
[charlist]	Any single character in charlist
[!charlist]	Any single character not in charlist

A group of one or more characters (charlist) enclosed in **brackets ([ ])** can be used to **match any single character in an expression** and can include almost any characters in the ANSI character set, including digits. In fact, the special characters left bracket ([ ), question mark (?), number sign (#), and asterisk (*) can be used to match themselves directly **only by enclosing them in brackets.** The right bracket ( ]) cannot be used within a group to match itself, but it can be used outside a group as an individual character. In addition to a simple list of characters enclosed in brackets, charlist can specify a **range of characters by using a hyphen (-)** to separate the upper and lower bounds of the range. For example, [A-Z] in pattern results in a match if the corresponding character position in expression contains any of the uppercase letters in the range A through Z. Multiple ranges are included within the brackets without any delimiting.

Examples

See Also:

Like Method

StartLine Property

Description:

The StartLine is of type longint and tells TVisAwk at which line to begin scanning a file.

Example:

```
VisAwk1.Filename := 'c:\readme.txt';
VisAwk1.Action := 1;
VisAwk1.ConvertCase := ccUpper;
VisAwk1.StartLine := 4;
VisAwk1.EndLine := 5;
VisAwk1.Execute;
```

The above example scans the file 'c:\readme.txt' and converts each line to upper case before parsing. Awk starts the scan at line 4 and ends at line 5.

See Also:

EndLine Property

TVisAwkMethods

Execute
GetItemPos
GetToken
Like

OutputEmbraceChars Property

Description:

Use the OutputEmbraceChars if you need to surround parsed tokens with a particular set of characters when creating an output file. You might want, for example, to surround the tokens with parentheses, so the setting of this property would be '()'. This property is generally useful only when creating an export file for another application that expects field contents to be embraced with certain characters.

See Also:

OutputSeparator Property

OutputFile Property

Description:

If you want to create an output file of the parsed tokens from a given file scan operation, simply set the outputfile property to the fully qualified path and file name of the file you want to create. If this property is anything other than an empty string TVisAwk will attempt to create the output file, there is no special action you as the developer need to take, it is done automatically when this property is set and the execute method is called.

Warning:

If the OutputOverwrite property is set to true then any existing file that matches the setting of the outputfile property will be overwritten. If the
OutputOverwrite property is set to false then the file will not be created.
You should probably supplement this functionality in your code by testing for the existence of the file and display a dialog to the user before calling the execute method.

See Also:

OutputOverwrite Property

StepMode Property

Description:

If you are using the visual display to display the status of a file parsing activity, you can set the StepMode property to true if you want to allow the user to step through the file one line at a time. If the value of the Visuals property is false, this property value is ignored.

See Also:

Visuals Property
KeepVisuals Property

Visuals Property

Description:

The Visual property is a boolean that determines whether or not to show a visual display of the parsing activity when scanning files. The visuals form contains five buttons which allow the user to interact with the parsing process but they are only visible if the KeepVisuals Property is true.

Done button - If the user clicks this button during file scan activity, the process will be aborted and the display removed.

Pause button - The user can click this button to pause the scan at the current line in the file.

Resume button-The resume button is enabled after the pause button is clicked and would normally be used when the user wants to resume scanning the file.

Step button - When the StepMode property is true, the Step button is enabled allowing the user to step through the file one line at a time.

Replay button - After a file scan is completed, the replay button is enabled allowing the user to click it to replay the entire operation.

The screen location of the visual display form can be altered by setting the VisualLeft and VisualTop properties prior to calling the execute method.

See Also:

KeepVisuals Property
StepMode Property
VisualLeft Property
VisualTop Property
Execute Method

VisualCaption Property

Description:

The VisualCaption property can be set to any string you want to use for the caption of the visual display form.

Example:

```
VisAwk1.VisualCaption := 'Parsing file';
```

VisualLeft Property

Description:

The VisualLeft property can be used to relocate the visual display form on the screen and is the value of that form's left property. This property should be set before calling the execute method.

See Also:

VisualTop Property

VisualTop Property

Description:

The VisualTop property can be used to relocate the visual display to a different part of the screen and represents the value of the left property of the visual display form. This property should be set before calling the execute method.

See Also:

VisualLeft Property

OutputOverwrite Property

Description:

OutputOverwrite is a boolean property that indicates whether TVisAwk should overwrite existing files when creating an output file of parsed tokens. You should take care when setting this property to true. If this property is set to false, TVisAwk will not create the output file when a matching file already exists. Add code to your application that tests for the existence of a file if you need to have the user confirm an overwrite action.

See Also:

Outputfile Property

OutputSeparator Property

Description:

When creating an output file of parsed tokens you will normally want to separate the tokens with some character, perhaps a comma. In this case the setting of this property would be ',' . This can be useful in creating comma-separated files for exporting data to another application.

See Also:

OutputEmbraceChars Property

LineNumber Property

Description:

The linenumber property contains the current line number that is being scanned in a file operation. It can be examined in the OnScan event handler to determine which line TVisAwk is currently working on.

See Also:

OnScan Event

GetItemPos Method

Description:

Use the GetItemPos method to retrieve the position (token number) of a given string within a line of text.

Example:

```
StrPcopy(VisAwk1.F, 'one two three');
VisAwk1.Action := 0;
VisAwk1.Execute;
i := VisAwk1.GetItemPos('two');
```

In the above example, i would result in 2, the second token in the line.
If the string passed to this method does not exist in the line, GetItemPos returns -1.

Like Method

Description:

The Like method is a function that returns true or false, the syntax is:

```
If VisAwk1.Like('sometext') then
...
```

The text string passed to the Like method is compared to the pattern contained in the Pattern property to determine if there is a match. The compare is case sensitive unless the ConvertCase property is set to either ccUpper or ccLower. If you want to scan a file for text that matches a pattern then set the Action property to 3, set the Pattern property to the desired pattern and then call the execute the method.

Example:

```
VisAwk1.Filename := 'c:\delphi\lib.pas');
VisAwk1.Pattern := 'function *isdigit*';
VisAwk1.Action := 3;
VisAwk1.execute;
```

This example scans the file for a function isdigit. Note that you do not have to call the Like method explicitly when the action property is set to 3. You would call it explicitly when you want to determine if a string matches a pattern, for example to explicitly test a string against a pattern:

```
VisAwk1.Pattern := 'borland*';
If VisAwk1.Like('borland international') then
...
```

See Also:

ConvertCase Property
Pattern Property
Pattern Matching Examples
OnPatternMatched Event

TVisAwk Events

OnBegin Event
OnEnd Event
OnError Event
OnParse Event
OnPatternMatched Event
OnScan Event

OnPatternMatched Event

Description:

The OnPatternMatched Event is fired when the text of the current line in a file scan operation matches the pattern specified in the pattern property.

No parameters are passed in the event handler, however, you can examine
the value of the F property to obtain the line of text which matched the pattern.

Note: Calling the Like method explicitly to test a string against a pattern does not cause this event to be fired since the like method returns a boolean value that may be used to test the result.

Example:

```
procedure TForm1.searchbuttonclick(Sender:TObject);
begin
   VisAwk1.Action := 3;
   VisAwk1.Filename := 'astronot.txt';
   VisAwk1.Pattern := '*the hubble telescope was deployed on*';
   VisAwk1.Execute;
end;

procedure VisAwk1.OnPatternMatched;
begin
   deploydate := StrPas(VisAwk1.FF);
   VisAwk1.Action := 2;   (* stop scanning and close file *)
end;
```

OnScan Event

Description:

VisAwk fires the OnScan event after scanning a line of text from a file.

Example:

```
Procedure TForm1.VisAwk1Scan;
begin
        Panel1.caption := 'Awk Scanning Line: ' + inttostr(Awk1.LineNumber);
end;
```

The above example uses the OnScan event to report the current line number. To obtain the entire line that was scanned you can examine the F property. Example:

```
Procedure TForm1.VisAwk1.Scan;
begin
      Panel1.caption := StrPas(Awk1.F); {shows the last line scanned}
end;
```

OnBegin Event

Description:

The OnBegin event is fired once before TVisAwk begins scanning a file.

Example:

```
procedure VisAwk1Begin;
begin
      MessageDlg('Starting file scan',mtInformation,[mbok],0);
end;
```

OnEnd Event

Description:

The OnEnd event is fired when TVisAwk finishes scanning a file.

Example:

```
procedure VisAwk1End;
begin
      MessageDlg('File processing completed.',mtInformation,[mbok],0);
end;
```

OnParse Event

Description:

The OnParse event is fired just before TVisAwk begins parsing a line of text.

Example:

```
Procedure Awk1Parse;
begin
        Panel1.caption := 'Parsing Line Number : ' +inttostr(Awk1.LineNumber)
end;
```

OnError Event

Description:

The OnError event is fired when Awk encounters an error. The errnum and errtext parameters can be used to diagnose the error.

Example:

```
Procedure VisAwk1Error(Errnum: integer; Errtext: string);
begin
        MessageDlg('Awk Error: ' + Inttostr(Errnum) + ' ' +  Errtext,mtWarning,[mbok],0);
end;
```

Note: You can also obtain the same information from the errornumber and errortext properties.

Pattern Matching Examples

The following examples show how you can use the like method Like to test expressions for different patterns. **(Note: maximize this window for best viewing).**

Kind of matching	pattern	returns True	returns False
Multiple characters	a*a	'aa', 'aBa', 'aBBBa'	'aBC'
Special character	a[*]a	'a*a	' 'aaa'
Multiple characters	ab*	'abcdefg','abc'	'cab', 'aab'
Single character	a?a	'aaa', 'a3a', 'aBa'	'aBBBa'
Single digit	a#a	'a0a', 'a1a', 'a2a'	'aaa', 'a10a'
Range of chars	[a-z]	'f', 'p','j'	'2','&'
Outside a range	[!a-z]	'9','&', '%'	'b', 'a'
Not a digit	[!0-9]	'A', 'a', '&', '~'	'0', '1', '9'
Combined	a[!b-m]#	'An9', 'az0', 'a99'	'abc', 'aj0'

TVisAwk Shareware Version

TVisAwk is shareware. You can use TVisAwk for 30 days to evaluate its suitability for use in developing your applications. If you use it after the 30 day period, you must register your copy. You may not create an application that uses the component without registering your copy of TVisAwk.

Benefits of Registration

Registered users of TVisAwk get ***free technical support*** via email, ***automatic notification*** of new updates and ***full source code***. You may use or modify the source code for whatever purpose you desire.

Source Code Licensing Restrictions:

You ***may not*** distribute or sell the source code to others. You ***may not*** use the source to build a component which descends from TVisAwk for commercial or free distribution. You ***may*** modify the visual display form in any way you deem necessary in developing your applications.

Technical Support

If you have problems, questions or comments about TVisAwk you can contact me at:

compuserve: 76350,3301
internet: jcta@mindspring.com
web page: www.mindspring.com/~jcta

Please include as much detail as possible in your correspondence if you have problems with the operation of the TVisAwk component or if you suspect a bug.

Registering by Credit Card

I have contracted with Northstar Solutions to process registrations that you wish to make with your Visa or MasterCard. **They should be contacted for registrations only.** They cannot provide technical support for TVisAwk nor can they answer questions about the functionality of the component.

To register by credit card have your credit card number and expiration date handy and contact Northstar Solutions at:

Voice: 1-800-699-6395 (10:00 am - 10:00 pm Eastern Standard Time - **U. S. callers only**)

1-803-699-6395 (10:00 am - 10:00 pm Eastern Standard Time - **International callers**)

FAX: 1-803-699-5465 (available 24 hours).

EMAIL: America Online: STAR MAIL Compuserve: 71561,2751
Internet: starmail@aol.com

Tell them you want to register John Taylor's TVisAwk component for Delphi.
They notify me daily of registrations and as soon as they've contacted me you will receive via email (or compuserve if you're a member) the full distribution including the source code. If you do not have email capability and you require a diskette, a $ 5 (US) surcharge will be applied for postage and handling.

Copyright and Disclaimer

Disclaimer:

TVisAwk contains not warranties either express or implied as to its suitability for a particular purpose. You have 30 days to try TVisAwk to determine for yourself if it suits your needs.