System API Low-level Routines

Modified by Rob Evans on 15-Feb-2017

Parent page: Technical Reference - System API

System API: Low Level Routines Reference

Contents of this reference:

Scale Factor Table

T 1012
G 109
M, Meg = 106
K 103
U 10-6
N 10-9
P 10-12
F 10-15

Constants

```
cMeasureUnitSuffixes : Array[TMeasureUnit] Of TDynamicString = ('mil', 'mm',
'in', 'cm', 'dxp', 'm');
cMeasureUnitConvert  : Array[TMeasureUnit, TMeasureUnit] Of Double =
(// to  mil           mm         in        cm          dxp         m
        (1          , 2.54/100 , 1/1000  , 2.54/1000 , 1/10      ,
2.54/100000), // from mils
        (100/2.54   , 1        , 1/25.4  , 1/10      , 10/2.54   , 1/1000
), // from mm
        (1000       , 25.4     , 1       , 2.54      , 100       , 0.0254
), // from in
        (1000/2.54  , 10       , 1/2.54  , 1         , 100/2.54  , 1/100
), // from cm
        (10         , 2.54/10  , 1/100   , 2.54/100  , 1         , 2.54/10000
```

```
), // from dxp
        (100000/2.54, 1000     , 100/2.54, 100       , 10000/2.54, 1
)  // from m
);

cPaintColorModes : Array[TPaintColorMode] Of TDynamicString = ('FullColor',
'GrayScale', 'Monochrome');

  CaseSensitive   = True;
  CaseInSensitive = False;
  OrdNumOfZero    = 48;
  cDefThumbnailSizeX = 96;
  cDefThumbnailSizeY = 72;

   Delimiter       : Set of char = [#0,#39,',',' ',#10,#13,#9,'(',')'];
   StringDelimiter = #39;

  cm_Share_Compat     = $0;
  cm_Share_DenyRW     = $10;
  cm_Share_DenyW      = $20;
  cm_Share_DenyR      = $30;
  cm_Share_DenyN      = $40;
  cm_Access_ReadOnly  = $0;
  cm_Access_WriteOnly = $1;
  cm_Access_ReadWrite = $2;
  cm_NoInheritance    = $80; {A child process would not inherit file handle
and mode}

  fe_NoAccessError                = $0;
  fe_FunctionInvalid              = $1;
  fe_FileNotFound                 = $2;
  fe_PathNotFoundOrFileDoesntExist = $3;
  fe_NoHandleIsAvalible           = $4;
  fe_AccessIsDenied               = $5;
  fe_FileAccessCodeInvalid        = $C;

   FileExtension_Temp         = '$$$';

   cPathSeparator         = '\';

    cBooleanStrings : Array[False..True] Of TString = ('False','True');
```

Conversion Routines

```
Function GetPrevSettings_Count : Integer;
Function GetPrevSettings_Name                                (AIndex :
Integer) : TDynamicString;
Function GetPrevSettings_SpecialKey_SoftwareAltiumApp        (AIndex :
Integer) : TDynamicString;
Function GetPrevSettings_SpecialKey_SoftwareAltiumAppDXP     (AIndex :
```

```
Integer) : TDynamicString;
Function GetPrevSettings_SpecialFolder_AltiumApplicationData (AIndex :
Integer) : TDynamicString;

Function ConvertMeasureUnits(Const AValue : Double; FromUnit, ToUnit :
TMeasureUnit) : Double;

Function StripMeasureUnits(Var S : TDynamicString; Var Value : Double; Var
UsedUnits : TMeasureUnit) : Boolean;
```

Enumerated Types

TAltShiftCtrlCombination

```
TAltShiftCtrlCombination = TShiftState;
```

TBoolean

```
TBoolean       = Boolean;
```

TBusKind

```
TBusKind           =
(eBusKindUndefined,eBusKindLowValueFirst,eBusKindHighValueFirst,eBusKindGener
ic);
```

TByte

```
TByte          = Byte;
```

TChar

```
TChar  = Array[0..256] of Char;
```

The Char type is equivalent to AnsiChar. Because the implementation of Char is subject to change, it's a good idea to use the standard function SizeOf rather than a hard-coded constant when writing programs that may need to handle characters of different sizes. The TChar type can be used instead of a PChar.

Example

```
Var
  P : TChar;
Begin
    lResult := GetModuleFileName(HInstance,P,255)
....
End;
```

TDate

```
TDate = Record
```

```
    Year   : Word;
    Month  : Word;
    Day    : Word;
End;
```

TDouble

```
TDouble = Double;
```

TDynamicString

```
TDynamicString = AnsiString;
```

TExtended

```
TExtended      = Extended;
```

TFileFunction

```
TFileFunction = Function(Path : TDynamicString) : Boolean Of Object;
```

THugeInt

```
THugeInt       = Comp;
```

TMatchFileNameKind

```
TMatchFileNameKind = (eMatchByPath,eMatchByFileName);
```

TPaintColorMode

```
TPaintColorMode    = (ePaintColorMode_FullColor, ePaintColorMode_GrayScale,
ePaintColorMode_Monochrome);
```

TMeasureUnit

```
TMeasureUnit = (cUnitMil, cUnitMM, cUnitIN, cUnitCM, cUnitAltium Designer,
cUnitM);
```

TPaintScaleMode

```
TPaintScaleMode = (psmScreen, psmDefault, psmPrint);
```

TReal

```
TReal          = Single;
```

TString

```
TString = ShortString;
```

TTime

```
TTime = Record
    Hours        : Word;
```

```
    Minutes      : Word;
    Seconds      : Word;
    MilliSeconds : Word;
End;
```

TNonRefCountedInterfaceObject

```
TNonRefCountedInterfacedObject = Class(TObject, IInterface)
  Protected
    FRefCount : Integer;
    Function    QueryInterface(Const IID: TGUID; Out Obj): HResult; StdCall;
    Function    _AddRef: Integer;                                   StdCall;
    Function    _Release: Integer;                                  StdCall;
End;
```

Dialogs

ConfirmOkCancel

Declaration
`Function  ConfirmOKCancel (S : TDynamicString) : Boolean;`
Description
The ConfirmOkCancel function displays a dialog with the S parameter for the message body of the dialog. This function returns a Boolean value. Since there are ‘OK’ and ‘Cancel’ buttons, if you pressed the OK button, the functions returns a true value, otherwise the function returns a false value
See also
ConfirmNoYes, ShowError, ShowInfo, ShowWarning procedures.

ConfirmOkCancelWithCaption

Declaration
`Function  ConfirmOKCancelWithCaption   (Caption, S : TDynamicString) : Boolean;`
Description
The ConfirmOkCancelWithCaption function displays a dialog with a Caption parameter for the title bar of the dialog, and the S parameter for the message body of the dialog. This function returns a Boolean value. Since there are ‘OK’ and ‘Cancel’ buttons, if you pressed the OK button, the functions returns a true value, otherwise the function returns a false value
See also
ConfirmNoYes, ShowError, ShowInfo, ShowWarning procedures.

ConfirmNoYes

Declaration
`Function ConfirmNoYes(Const S: String) : Boolean`
Description
The procedure displays a message dialog with a YES button and NO button buttons. The title of the message box is "Confirm". The Value parameter returns True for the button Yes and False for no.
See also
Dialogs

ConfirmNoYesCancel

Declaration
```
Function ConfirmNoYesCancel(Const S: String) : Integer
```
Description
The procedure displays a message dialog with a YES button, NO button and Cancel buttons. The title of the message box is "Confirm".
The Value parameter returns one of the following values as a TModalResult type (as defined in Borland Delphi) representing which button has been pressed.
See also
ConfirmNoYes, ShowError, ShowInfo, ShowWarning procedures.

ConfirmNoYesCancelWithCaption

Declaration
```
Function  ConfirmNoYesCancelWithCaption(Const Caption, S : TDynamicString) : Integer;
```
Description
The ConfirmNoYesCancelWithCaption function displays a dialog with a Caption parameter for the title bar of the dialog, and the S parameter for the message body of the dialog and has ‘Yes’, ‘No’ and ‘Cancel’ buttons.
This function returns a modal value, ie when the user chose the Cancel button, an IDCancel (2) is returned or when the user chose the No button an IDNo (7) is returned, or when the user chose the Yes button, an IDYES (6) value is returned.
See also
ConfirmNoYes, ShowError, ShowInfo, ShowWarning procedures.

ConfirmNoYesWithCaption

Declaration
```
Function  ConfirmNoYesWithCaption      (Caption   : TDynamicString; S : TDynamicString) : TBoolean;
```
Description
The ConfirmNoYesWithCaption function displays a dialog with a Caption parameter for the title bar of the dialog, and the S parameter for the message body of the dialog and has ‘Yes’ and ‘No’ buttons.
This function returns a modal value, ie when the user user chose the No button a False value is returned, or when the user chose the Yes button, a True value is returned
See also
ConfirmNoYes, ShowError, ShowInfo, ShowWarning procedures.

SortedListBoxCompare

Declaration
```
Function SortedListBoxCompare(Const S1, S2 : AnsiString) : Integer;
```
Description
This function has its internal sorting routine that sorts lists alphanumerically. Delphi’s sort can only provide digital or alphabet sorting only. You will need to invoke the CustomSort routine for a TStringList or other Delphi equivalent string lists and pass this function into this CustomSort routine.
Example

See also

DisplayNotImplementedMessage

Declaration
```
Procedure DisplayNotImplementedMessage(ProcessId,ProcessDescription :
TDynamicString);
```
Description
This procedure displays a dialog with the Server Process not Implemented Message for server projects. This is used in the commands unit of a server project.
See also
ShowInfo and ShowWarning procedures.

RunNetworkConnectionDialog

Syntax
Procedure RunNetWorkPrintersDialog(HWindow : Hwnd);
Description
This procedure invokes the Network Printers dialog with the handle of the current dialog or window in Altium Designer.
Example

See also

RunNetworkPrintersDialog

Syntax
Procedure RunNetWorkConnectionDialog(HWindow : Hwnd);
Description
This procedure invokes the Network Connection dialog with the handle of the current dialog or window in ALTIUM DESIGNER.
Example

See also

RunOpenDocumentDialog

Syntax
```
Function  RunOpenDocumentDialog (Caption : TDynamicString; MultiSelect :
Boolean; Var Path, SelectedType, Editor : TDynamicString;  Const FileTypes,
Files : TStrings) : Boolean;
```
Description
This function is based on the Client's `RunCommonDialog` process. The `Caption` parameter is used for the Title of the dialog. The `MultiSelect` parameter allows you to select files from the dialog if True. If you want to only select one file use the `False` value. The `Path`, `SelectedType` and `Editor` parameters are returned after the dialog has closed. FileTypes and Files parameters determine which file types and files can be opened by the Common Dialog.
Example

See also

ShowError

Declaration

```
Procedure ShowError(Const S: String);
```
Description
This procedure displays an Error dialog containing an OK button and the warning icon.
See also
ShowInfo and ShowWarning procedures.

ShowError_SystemModal

Syntax
```
Procedure ShowError_SystemModal(Const S : TDynamicString);
```
Description
The `ShowError_SystemModal` procedure displays an independent dialog with an error symbol and string, S, for the text. This dialog does not have the Altium Designer's window handle and thus appears on the taskbar of the Windows Desktop.
Example

See also

ShowInfo

Declaration
```
Procedure ShowInfo(Const S: String);
```
Description
The procedure displays an information dialog containing an OK button and the information icon.
See also
ShowError and ShowWarning procedures.

ShowInfoWithCaption

Declaration
```
Procedure ShowInfoWithCaption          (Caption,S : TDynamicString);
```
Description
Displays a dialog with the Information icon and with a Caption parameter for the title bar of the dialog, and the S parameter for the message body of the dialog.
See also
ShowError and ShowWarning procedures.

ShowWarning

Declaration
```
Procedure ShowWarning(Const S: String);
```
Description
This procedure displays a warning dialog containing an OK button and the warning icon.
See also
ShowError and ShowInfo procedures.

File IO

AddBackSlashToFrontAndBack

Declaration
```
Function  AddBackSlashToFrontAndBack(S: TDynamicString) : TDynamicString;
```

Description
The AddBackSlashToFrontAndBack function adds a path separator character to the front and to the back of a string. For example if the S string is empty, only one back slash is added to the string. Otherwise the S string has a back slash added to the front and to the end of this string.
See also

CheckAgainstWildCard_CaseSensitive

Declaration
```
Function  CheckAgainstWildCard_CaseSensitive(WildCard,Name : TDynamicString)
```
Description
The CheckAgainstWildCard_CaseSensitive function allows the comparison of the Wildcard string containing wildcards to the Name string. Use the Wildcard string which can consist of upper case and lower case characters to determine if the Name string matches the format described by the Wildcard string. The wildcard string can contain wildcards that can match any character, and sets that match a single character that is included in the Name string.
See also

CheckAgainstWildCard

Declaration
```
Function  CheckAgainstWildCard (WildCard,Name : TDynamicString)
```
Description
The CheckAgainstWildCard function allows the comparison of the Wildcard string containing wildcards to the Name string. Use the Wildcard string to determine if the Name string matches the format described by the Wildcard string. The wildcard string can contain wildcards that can match any character, and sets that match a single character that is included in the Name string. This function is not case sensitive.
See also

ComputerName

Declaration
```
Function ComputerName : ShortString
```
Description
The ComputerName function retrieves the computer name of the current system. This name is established at system startup, when it is initialized from the registry.
See also

ConvertDiskSizeToString

Declaration
```
Function  ConvertDiskSizeToString    (Size : Integer) : TDynamicString;
```
Description
The ConvertDiskSizeToString function converts a number into a string representing the size of a storage space. For example, when Size = 345, then the function returns a ‘345 Bytes’ string.
See also

ConvertFIleNameToExeSystemFileName

Declaration
```
Function ConvertFileNameToExeSystemFileName(S : TString) : TString;
```
Description

The ConvertFileNameToExeSystemFileName routine updates the file name to include the full path to Altium\System folder along with the filename parameter. An example is 'C:\Program Files\Altium\System\ServerA.exe'
Example

See also

ConvertPartialPathToExeFileName

Delaration
```
Function ConvertPartialPathToExeFileName(S : TString) : TString;
```
Description
The ConvertPartialPathToExeFileName routine updates the file name to include the full path to Altium\System folder along with the filename parameter. An example is 'C:\Program Files\Altium\System\ServerA.exe'
Example

See also

CurrentModuleName

Syntax
```
Function CurrentModuleName : TString;
```
Description
The CurrentModuleName function retrieves the full path and filename for the executable/dynamic library linking file containing the specified module.
Example

See also

DocumentIsReadOnly

Declaration
```
Function DocumentIsReadOnly      (FullPath : TDynamicString) : Boolean;
```
Description
The DocumentIsReadOnly function returns True if a design document file has a read only property set true.
Example

```
If DocumentIsReadOnly(Document.FileName) Then
Begin
    ShowError(ExtractFileName(Document.FileName) + ' is read-only.');
    Exit;
End;
```

See also
ExtractFilename function

ExistAnyWhere

Declaration
```
Function ExistAnyWhere(Var S : TDynamicString) : TBoolean; Overload;
```

```
Function ExistAnyWhere(Var S : TString       ) : TBoolean; Overload;
```
Description
The ExistAnyWhere function returns a TBoolean value denoting whether the file exists or not. Note that the S parameter is of TDynamicString type.
Example

```
// Remove the .SchLib file because it is no longer needed
SchLibFileName := GetProjectLibraryPath;
If ExistAnyWhere(SchLibFileName) Then
Begin
    Project.DM_RemoveSourceDocument(SchLibFileName);
    Document := Client.GetDocumentByPath(SchLibFileName);
    If Document <> Nil Then Document.ReleaseFileOwnership;
    DeleteFile(SchLibFileName);
End;
```

See also
ExistAnyWhereAsTemplate function

ExistAnyWhereAsTemplate

Declaration
```
Function ExistAnyWhereAsTemplate(Var S : TDynamicString) : TBoolean;
```
Description
Checks if the S parameter containing the filename exists in the following folders:
SpecialFolder_DesignTemplates,
SpecialFolder_AltiumSystemTemplates,
SpecialFolder_TemplatesForAllUsers, or
SpecialFolder_CommonDocumentTemplates.
Example
```
If Not ExistAnywhere(MacroFileName) then Exit;
```
See also
ExistAnyWhere function.

ExpandFile

Declaration
```
Function ExpandFile (S : TDynamicString) : TDynamicString;
```
Description
The ExpandFile function converts the relative file name into a fully qualified path name by merging in the current drive and directory. A fully qualified path name includes the drive letter and any directory and sub-directories in addition to the file name and extension.
The ExpandFileName function does not verify that the resulting fully qualified path name refers to an existing file, or even that the resulting path exists.
Example
ShowMessage(ExpandFileName('autoexec.bat'));
See also
ExtractFilename function
FileExists function

FindFileAnyWhere

Declaration
```
Function FindFileAnyWhere(Var Path : TDynamicString) : TBoolean; Overload;
```
Description
This `FindFileAnywhere` checks if the file exists in the path or anywhere else. If a file is found, a 'True' value is returned, otherwise, 'False'
Example

See also

FileExists

Declaration
```
Function FileExists(const FileName: string): Boolean;
```
Description
The `FileExists` function returns True if the file specified by FileName exists. If the file does not exist, FileExists returns False.
Example

```
Function OpenProject(ProjectName : String) : Boolean;
Begin
    Result := True;
    If Not FileExists(ProjectName) Then Result := False;

    ResetParameters;
    AddStringParameter('ObjectKind','Project');
    AddStringParameter('FileName', ProjectName);
    RunProcess('WorkspaceManager:OpenObject');
End;
```

See also

GetFreeDiskSpaceString

Declaration
```
Function  GetFreeDiskSpaceString(DiskNumber : Integer) : TDynamicString;
```
Description
The `GetFreeDiskSpaceString` function returns a TDynamicString value which represents the number of free bytes on the specified drive number.
See also

GetDiskSizeString

Declaration
```
Function  GetDiskSizeString     (DiskNumber : Integer) : TDynamicString;
```
Description
The `GetDiskSizeString` function returns a TDynamicString value which represents the size, in bytes, of the specified drive.
See also

GetDiskFree

Declaration
`Function GetDiskFree(Drive: Byte): Double;`
Description
The `GetDiskFree` function returns a double value which reports the amount of free space on the disk. The Drive value (Byte value) represents the drive letter. A drive = 0, B Drive = 1 etc.
See also

GetMacroDescription

Declaration
Function `GetMacroDescription`(MacroFileName : TString) : TString;
Description
This GetMacroDescription returns a string if the function finds the '$SUMMARY' or '$Description' identifier in a macro script.
Example

See also

HasExtension

Declaration
`Function HasExtension(Const Name : TDynamicString; Var DotPos : Integer) : TBoolean;`
Description
This function checks if the Name parameter has an extension by scanning for the dot character. If the dot character is found, the index of the DotPos variable parameter is returned. Note that the invalid characters are '\' and ':' and if they exist in the Name parameter, then the function returns a false value.
See also

IsFullPathToExistingFile

Declaration
`Function IsFullPathToExistingFile(FullPath : TDynamicString) : Boolean;`
Description
This function returns True if the path including the filename to an existing file exists. Use this function to distinguish a path that contains the filename only.
See also
IsFullPathToExistingStructuredStorage function

IsFullPathToExistingStructuredStorage Function

Declaration
`Function IsFullPathToExistingStructuredStorage(Const FullPath : TDynamicString) : Boolean;`
Description
This function indicates whether a particular disk file contains a storage object. This function returns True if the path including the filename to an existing structured storage exists.
Example

```
If IsFullPathToExistingStructuredStorage(GetFileName) Then
    Result := fmShareDenyNone
Else
    Result := Inherited GetFileShareMode;
```

See also
IsFullPathToExistingFile function

IsPathRelative

Declaration
```
Function IsPathRelative(Path : TString) : Boolean;
```
Description
This `IsPathRelative` function checks if the string contains a relative path not a full absolute path.
Example

```
    If IsPathRelative(FileName) Then
    Begin
        If Not DirectoryExists(FRootPath) Then Exit;

        S := GetCurrentDir;
        If Not SetCurrentDir(FRootPath) Then Exit;
        Try
            AbsolutePath := ExpandFileName(FileName);
        Finally
            SetCurrentDir(S);
        End;
    End
    Else
        AbsolutePath := FileName;
```

See also
ExpandFilename function

LowLevelRunTextEditorWithFile

Declaration
```
Procedure LowLevelRunTextEditorWithFile   (S : TDynamicString);
```
Description
This function invokes the Microsott Windows NotePad application and attempts to open the file denoted by the S parameter.
See also
RunCommand procedure.

ProcessAllFilesOnPath

Declaration
```
Procedure ProcessAllFilesOnPath(Filter           : TDynamicString;
                                FileFunction     : TFileFunction;
                                AbsolutePath     : TDynamicString;
                                IncludeSubFolders : Boolean = True);
```

Description
This function returns all files on the specified `AbsolutePath` and `Filter` parameters. Normally to fetch all files on the Absolute path, use this `'*'` `Filter` String. Note only one asterisk for the `Filter` parameter. Otherwise you can use the following filters for example, '*.*' and '*.Schlib'. The FileFunction parameter outputs strings in a `TStringList` object.
Example
```
ProcessAllFilesOnPath('*',ArchiveItems_CreateAnyDirectoryFile,AFullPath,True)
;
```
See also
TFileFunction type

ValidDosFileName

Declaration
```
Function ValidDosFileName(FileName : TSTring) : TBoolean;
```
Description
The `ValidDosFileName` returns a TBoolean value denoting whether the filename string is a valid DOS filename. A valid dos filename must not have the following characters ('*','?',' ','"','/',';' ,'|',',', '=') and only have one '.' fullstop character in the entire filename string.
Example

```
Filename := ForceFileNameExtension(Board.FileName, ReportFileExtension);
If GetState_ParameterUpperCaseString(Parameters, 'Filename', S) Then
    If (ValidDosFileName(S)) then Filename := S;
```

See also
ForceFileNameExtension function

Number Manipulation Routines

GetBinaryStringFromInteger

Declaration
```
Function  GetBinaryStringFromInteger(L : Integer   ) : TDynamicString;
```
Description
The `GetBinaryStringFromInteger` function converts an integer to a binary string (up to thirty two characters long). An integer contains 4 bytes = 32 bits.
See also

ExtendedToEng

Declaration
```
Function  ExtendedToEng (Const ExtVal    : Extended) : String;
```
Description
The `ExtendedToEng` function converts the floating-point value given by Value to its string representation.
Example
```
ShowInfo(ExtendedToEng(4.32e18)); //4.320e18
```
See also
Number Manipulation routines

EngToExtended

Declaration
```
Function  EngToExtended (Const EngString : String)   : Extended;
```
Description
The EngToExtended function converts the string value given by EngString to its extended representation. This function looks at the last character of the string and converts it accordingly - see scale factor table below. For example ‘3Meg’ will come out as 3M.
See also
Number Manipulation routines

GetHexStringFromInteger

Declaration
```
Function  GetHexStringFromInteger   (L : Integer) : TDynamicString;
```
Description
The `GetHexStringFromInteger` converts a word to a hexadecimal string (up to eight characters long). The hexadecimal number system is a base 16 system with 16 digits. A byte equals 2 hexademical digits because each hexadecimal digit corresponds to four binary digits thus 4 bytes equals 8 hexadecimal digits.
See also
Number Manipulation routines

HexToInteger

Declaration
```
Function HexToInteger(Const S : TDynamicString) : Integer;
```
Description
Convert a hexadecimal value (as a string value) to an Integer value.
See also
Number Manipulation routines

IntegerToHex

Declaration
```
Function IntegerToHex(L : Integer) : TDynamicString;
```
Description
Convert an integer value to an hexadecimal value.
See also
Number Manipulation routines

IntMax

Declaration
```
Function  IntMax(x,y : Integer) : Integer;
```
Description
The `IntMax` function returns the maximum value of X and Y integer types.
See also
Number Manipulation routines

IntMin

Declaration

```
Function  IntMin(x,y : Integer) : Integer;
```
Description
The `IntMin` function returns the minimum value of X and Y integer types.
See also
Number Manipulation routines

IntSwap

Declaration
```
Procedure IntSwap(Var a,b : Integer);
```
Description
The `IntSwap` procedure swaps the values for A and B. For example A = 2 and B = 5. After passing these values into IntSwap procedure, the new values are a = 5 and b = 2.
See also
Number Manipulation routines

Other Routines

AltKeyDown

Declaration
```
Function AltKeyDown: Integer;
```
Description
This function returns a value that indicates the state of the ALT key, that is, the function returns 1 if the ALT key is pressed down, otherwise it returns 0.
See also
Other Routines

BeginHourGlass

Declaration
```
Procedure BeginHourGlass(ACursor : TCursor = crHourGlass);
```
Description
The `BeginHourGlass` procedure changes the cursor to a Hour Glass that denotes that the system is busy.
See also
EndHourGlass procedure
SetCursorBusy procedure
Other Routines

CheckActiveServer

Declaration
```
Function CheckActiveServer(Const AServerName, AServerCaption: String;
AWithDialog: Boolean): Boolean;
```
Description
The function checks whether the server for the nominated document is active or not.
See also
Other Routines

ControlKeyDown

Syntax
```
Function ControlKeyDown: Integer;
```
Description
The ControlKeyDown function returns a value that indicates the state of the CONTROL key, that is, the function returns 1 if the CONTROL key is down, otherwise it returns 0.
See also
AltKeyDown and ShiftKeyDown functions.
Other Routines

BeginHourGlass

(ClientAPIReg unit)
Declaration
```
Procedure BeginHourGlass(ACursor : TCursor = crHourGlass);
```
Description
The EndHourGlass procedure changes the cursor from a Hour Glass cursor back to the default pointing cursor.
See also
BeginHourGlass procedure
SetCursorBusy procedure
Other Routines

EscKeyDown

Syntax
```
Function EscKeyDown: Integer;
```
Description
The EscKeyDown function returns a value that indicates the state of the ESCAPE key, that is, the function returns 1 if the ESCAPE key is down, otherwise it returns 0.
See also
AltKeyDown and ShiftKeyDown functions.
Other Routines

GetActiveServerName function

Syntax
```
Function GetActiveServerName:String;
```
Description
The GetActiveServerName function returns the name of the server module that is currently active in Altium Designer.
Example

See also
Other Routines

GetCurrentWindowHandle

Declaration
```
Procedure GetCurrentWindowHandle(Var Value: HWND);
```
Description

The procedure returns an HWND value which represent the window handle of the currently active window in Altium Designer.

See also

Other Routines

GetCurrentDocumentFileName

Declaration

```
Function GetCurrentDocumentFileName : String;
```

Description

The `GetCurrentDocumentFileName` obtains the filename of the currently focussed document in DXP.

See also

SaveCurrentDocument function.
Other Routines

GetErrorMessage

Declaration

```
Function GetErrorMessage(Const ErrorNumber : Integer) : String;
```

Description

The `GetErrorMessage` function returns an error message string that corresponds to the specified Operating System error code.

See also

Other Routines

RunApplication

Declaration

```
Function RunApplication(Const CommandLine : String) : Integer;
```

Description

The `RunApplication` function executes an application program outside the Altium Designer environment. You need to supply the full path including the filename to the application you wish to execute.

Example

```
CommandLine := 'notepad.exe' + NameOfTextFile;
ErrorCode   := RunApplication(CommandLine);
If ErrorCode <> 0 Then
    ShowError('System cannot start : ' + CommandLine + #13#10 +
GetErrorMessage(ErrorCode));
```

See also

Other Routines

ResetCursor

Declaration

```
Procedure ResetCursor;
```

Description

The `ResetCursor` resets the cursor to the default arrow cursor.

See also

SetCursorBusy
Other Routines

RunCommand

Syntax
```
Procedure RunCommand (Const IdString : TDynamicString; Const SpecialParameter
: TDynamicString);
```
Description
This procedure executes a server process with parameters. The `IdString` parameter denotes the `servername:serverprocessname`. The `SpecialParameter` parameter denotes the `parametername=parametervalue` blocks separated by the | pipe symbol.
This RunCommand function is not properly supported by the scripting system in Altium Designer.
Examples

```
RunCommand('Client:SetupPreferences', 'Server=PCB|PageName=Models');
RunCommand('WorkspaceManager:Configure','ObjectKind=MessageView|Action=ClearA
ll');
RunCommand('PCB:BoardInformation','');
RunCommand('PCB:Zoom','Action=Redraw');
```

See also
RunSystemCommand

RunSystemCommand

Syntax
```
Function RunSystemCommand(Const S : TDynamicString) : TBoolean;
```
Description
The RunSystemCommand function runs the specified application denoted by the parameter string, S.
Example
```
RunSystemCommand('NotePad.Exe ' + S);
```
See also
RunCommand procedure.

RunSystemCommandInSystemDirectory

Syntax
```
Function  RunSystemCommandInSystemDirectory(Const S : TDynamicString) :
TBoolean;
```
Description
The RunSystemCommandInSystemDirectory function runs the specified application in the Windows directory and the application’s filename is denoted by the string, S.
Example
```
RunSystemCommandInSystemDirectory(‘Notepad.Exe’);
```
See also
RunCommand procedure
RunSystemCommand procedure

SaveCurrentDocument

Syntax

```
Function SaveCurrentDocument : Boolean;
```
Description
The SaveCurrentDocument function determines whether the current document can be saved or not.
See also
Other Routines

SetCursorBusy

Declaration
```
Procedure SetCursorBusy;
```
Description
The SetCursorBusy updates the cursor to the default busy cursor, to indicate that the system is busy. This procedure could be set before a time consuming loop within a script.
See also
ResetCursor
Other Routines

ShiftKeyDown

Declaration
```
Function ShiftKeyDown: Integer;
```
Description
The ShiftKeyDown function returns a value that indicates the state of the SHIFT key, that is, the function returns 1 if the SHIFT key is down, otherwise it returns 0.
See also
AltKeyDown and ControlKeyDown functions.
Other Routines

Special Folder Path Strings

panic.

ClientAPI_SpecialFolder_AltiumAllUserApplicationData

Syntax
```
Function ClientAPI_SpecialFolder_AltiumAllUserApplicationData : WideString;
```
Description
This function returns the full path to the special folder.
Example
ShowMessage(ClientAPI_SpecialFolder_AltiumAllUserApplicationData);
```
//C:\Documents and Settings\All Users\Application Data\AltiumDesigner
```
See also
Special Folder Paths

ClientAPI_SpecialFolder_AltiumApplicationData

Syntax
```
Function ClientAPI_SpecialFolder_AltiumApplicationData : WideString;
```
Description
This function returns the full path to the special folder.
Example

ShowMessage(ClientAPI_SpecialFolder_AltiumApplicationData);
//C:\Documents and Settings*UserName*\Application Data\AltiumDesigner
See also
Special Folder Paths

ClientAPI_SpecialFolder_AltiumLocalApplicationData

Syntax
Function ClientAPI_SpecialFolder_AltiumLocalApplicationData : WideString;
Description
This function returns the full path to the special folder.
Example
ShowMessage(ClientAPI_SpecialFolder_AltiumLocalApplicationData);
//C:\Documents and Settings*UserName*\Local settings\Application Data\AltiumDesigner
See also
Special Folder Paths

SpecialFolder_AdminTools

Declaration
Function SpecialFolder_AdminTools : TDynamicString;
Description
This function returns the path to the All User Application Data folder.
See also
Special Folder Paths

SpecialFolder_AllUserAdminTools

Declaration
Function SpecialFolder_AllUserAdminTools : TDynamicString;
Description
This function returns the path to the C:\Documents and Settings\All Users\Start Menu\Programs\Administrative Tools folder.
See also
Special Folder Paths

SpecialFolder_AllUserDesktop

Declaration
Function SpecialFolder_AllUserDesktop : TDynamicString;
Description
This function returns the path to the C:\Documents and Settings\All Users\Desktop folder.
See also
Special Folder Paths

SpecialFolder_AllUserDocuments

Declaration
Function SpecialFolder_AllUserDocuments : TDynamicString;
Description
This function returns the path to the C:\Documents and Settings\All Users\Desktop folder.
See also

Special Folder Paths

SpecialFolder_AltiumLibraryIntegrated

Declaration
```
Function SpecialFolder_AltiumLibraryIntegrated : TDynamicString;
```
Description
This function returns the path to the Altium Integrated Library folder. Example `C:\Program Files\Altium\Library\`
See also
Special Folder Paths

SpecialFolder_AltiumLibraryPld

Declaration
```
Function SpecialFolder_AltiumLibraryPld : TDynamicString;
```
Description
This function returns the path to the Altium PLD Library folder. Example `C:\Program Files\Altium\Library\Pld\`
See also
Special Folder Paths

SpecialFolder_AltiumLibrary

Declaration
```
Function SpecialFolder_AltiumLibrary : TDynamicString;
```
Description
This function returns the path to the Altium Library folder. Example `C:\Program Files\Altium Designer\Library\`
See also
Special Folder Paths

SpecialFolder_AltiumApplicationData

Declaration
```
Function SpecialFolder_AltiumApplicationData : TDynamicString;
```
Description
This function returns the path to the Altium User Application Data folder. Example `C:\Documents and Settings\UserName\Application Data\Altium`
See also
Special Folder Paths

SpecialFolder_AltiumAllUserApplicationData

Declaration
```
Function SpecialFolder_AltiumAllUserApplicationData  : TDynamicString;
```
Description
This function returns the path to the Altium All User Application Data folder. Example `C:\Documents and Settings\All Users\Application Data\Altium`
See also
Special Folder Paths

SpecialFolder_AltiumDesignExplorer

Declaration
```
Function SpecialFolder_AltiumDesignExplorer : TDynamicString;
```
Description
This function returns the path to the Altium folder. Example `C:\Program Files\Altium\`
See also
Special Folder Paths

SpecialFolder_AltiumLocalApplicationData

Declaration
```
Function SpecialFolder_AltiumLocalApplicationData : TDynamicString;
```
Description
This function returns the path to the Altium Local Application Data folder. Example `C:\Documents and Settings\UserName\My Documents\My Designs`
See also
Special Folder Paths

SpecialFolder_AltiumSystem

Declaration
```
Function SpecialFolder_AltiumSystem : TDynamicString;
```
Description
This function returns the path to the Altium's system folder. Example `C:\Program Files\Altium\System\`
See also
Special Folder Paths

SpecialFolder_AltiumSystemTasksPages

Declaration
```
Function SpecialFolder_AltiumSystemTasksPages : TDynamicString;
```
Description
This function returns the path to the Altium's system tasks pages folder. Example `C:\Program Files\Altium\System\`
See also
Special Folder Paths

SpecialFolder_AltiumSystemTemplates

Declaration
```
Function SpecialFolder_AltiumSystemTemplates : TDynamicString;
```
Description
This function returns the path to the Altium's System Templates folder. Example `C:\Program Files\Altium\System\Templates\`
See also
Special Folder Paths

SpecialFolder_AllApplicationData

Declaration
```
Function SpecialFolder_AllUserApplicationData : TDynamicString;
```

Description
This function returns the path to the `C:\Documents and settings\All Users\Application Data` folder.
See also
Special Folder Paths

SpecialFolder_AltiumTaskingApplicationData

Declaration
```
Function SpecialFolder_AltiumTaskingApplicationData : TDynamicString;
```
Description
This function returns the path to the Altium Tasking application data folder for example `C:\Documents and Settings\UserName\Application Data\Altium Designer\Tasking`.
See also
Special Folder Paths

SpecialFolder_AltiumSecurityAllUserApplicationData

Declaration
```
Function SpecialFolder_AltiumSecurityAllUserApplicationData : TDynamicString;
```
Description
This function returns the path to the Altium Security All User Application Data folder for example `C:\Documents and Settings\UserName\Application Data\AltiumDesignerSecurity\.`
See also
Special Folder Paths

SpecialFolder_AltiumSystemResources

Declaration
```
Function SpecialFolder_AltiumSystemResources : TDynamicString;
```
Description
This function returns the path to the Altium System Resources folder for example `C:\Program Files\Altium Designer\System\Resources`.
See also
Special Folder Paths

SpecialFolder_AltiumSystemDesktopLayouts

Declaration
```
Function SpecialFolder_AltiumSystemDesktopsLayouts : TDynamicString;
```
Description
This function returns the path to the Altium Device Images folder.
See also
Special Folder Paths

SpecialFolder_AltiumHelp

Declaration
```
Function SpecialFolder_AltiumHelp : TDynamicString;
```
Description
This function returns the path to the Altium Help folder for example `C:\Program Files\Altium Designer\System\Help\`
See also

Special Folder Paths

SpecialFolder_AltiumLocalResources

Declaration
```
Function SpecialFolder_AltiumLocalResources : TDynamicString;
```
Description
This function returns the path to the Altium Local resources folder for example `C:\Program Files\Altium Designer \System\`.
See also
Special Folder Paths

SpecialFolder_AltiumLocalHelp

Declaration
```
Function SpecialFolder_AltiumLocalHelp : TDynamicString;
```
Description
This function returns the path to the Altium Local help folder for example `C:\Program Files\Altium Designer\System\Help\`.
See also
Special Folder Paths

SpecialFolder_AltiumScripts

Declaration
```
Function SpecialFolder_AltiumScripts : TDynamicString;
```
Description
This function returns the path to the Altium Scripts folder for example `C:\Program Files\Altium Designer\Scripts\.`
See also
Special Folder Paths

SpecialFolder_AltiumSystemButtons

Declaration
```
Function SpecialFolder_AltiumSystemButtons : TDynamicString;
```
Description
This function returns the path to the Altium System Buttons folder for example `C:\Program Files\Altium Designer\System\Buttons\`.
See also
Special Folder Paths

SpecialFolder_AltiumSystemDocumentImages

Declaration
```
Function SpecialFolder_AltiumSystemDocumentImages : TDynamicString;
```
Description
This function returns the path to the Altium System Document Images folder for example `C:\Program Files\Altium Designer\System\DocumentImages\`.
See also
Special Folder Paths

SpecialFolder_AltiumSystemNavImages

Declaration
```
Function SpecialFolder_AltiumSystemNavImages : TDynamicString;
```
Description
This function returns the path to the Altium System Nav Images folder for example `C:\Program Files\Altium Designer\System\NavImages\`.
See also
Special Folder Paths

SpecialFolder_AltiumSystemNavPages

Declaration
```
Function SpecialFolder_AltiumSystemNavPages : TDynamicString;
```
Description
This function returns the path to the Altium System Nav Pages folder for example `C:\Program Files\Altium Designer\System\NavPages`.
See also
Special Folder Paths

SpecialFolder_AltiumLibraryVHDL87

Declaration
```
Function SpecialFolder_AltiumLibraryVHDL87 : TDynamicString;
```
Description
This function returns the path to the Altium Library VHDL 87 folder for example `C:\Program Files\Altium Designer\Library\VHDL\IEEE87\.`
See also
Special Folder Paths

SpecialFolder_AltiumLibraryVHDL93

Declaration
```
Function SpecialFolder_AltiumLibraryVHDL93 : TDynamicString;
```
Description
This function returns the path to the Altium Library VHDL93 folder for example `C:\program files\Altium Designer\library\VHDL\IEEE93\`.
See also
Special Folder Paths

SpecialFolder_AltiumLibraryVerificVHDL87

Declaration
```
Function SpecialFolder_AltiumLibraryVerificVHDL87 : TDynamicString;
```
Description
This function returns the path to the Altium Library Verific VHDL87 folder for example `c:\program files\Altium Designer\library\VHDL\VHDL87\.`
See also
Special Folder Paths

SpecialFolder_AltiumLibraryVerificVHDL93

Declaration

```
Function SpecialFolder_AltiumLibraryVerificVHDL93 : TDynamicString;
```
Description
This function returns the path to the Altium Library Verific VHDL93 folder for example `c:\program files\Altium Designer\library\VHDL\VHDL93\`.
See also
Special Folder Paths

SpecialFolder_AltiumSynthesis

Declaration
```
Function SpecialFolder_AltiumSynthesis : TDynamicString;
```
Description
This function returns the path to the Altium Synthesis folder, for example `c:\program files\Altium Designer\library\VHDL_LIB\`
See also
Special Folder Paths

SpecialFolder_AltiumLibraryEDIF

Declaration
```
Function SpecialFolder_AltiumLibraryEDIF : TDynamicString;
```
Description
This function returns the path to the Altium Library EDIF folder for example `c:\program files\Altium Designer\library\EDIF\`.
See also
Special Folder Paths

SpecialFolder_AltiumLibraryVHDL

Declaration
```
Function SpecialFolder_AltiumLibraryVHDL : TDynamicString;
```
Description
This function returns the path to the Altium Library VHDL folder for example `c:\program files\Altium Designer\library\VHDL\`.
See also
Special Folder Paths

SpecialFolder_AltiumLibraryVHDLModels

Declaration
```
Function SpecialFolder_AltiumLibraryVHDLModels : TDynamicString;
```
Description
This function returns the path to the Altium Library VHDL Models folder for example `c:\program files\Altium Designer\library\VHDL\Models\`.
See also
Special Folder Paths

AltiumLibraryLMF

Declaration
```
Function SpecialFolder_AltiumLibraryLMF : TDynamicString;
```
Description
This function returns the path to the Altium Library LMF folder for example `c:\program`

files\Altium Designer\library\EDIF\.
See also
Special Folder Paths

SpecialFolder_AltiumConstraintFiles

Declaration
```
Function SpecialFolder_AltiumConstraintFiles : TDynamicString;
```
Description
This function returns the path to the Altium Constraint Files folder for example `c:\program files\Altium Designer\library\FPGA\`.
See also
Special Folder Paths

SpecialFolder_AltiumDeviceConstraintFiles

Declaration
```
Function SpecialFolder_AltiumDeviceConstraintFiles : TDynamicString;
```
Description
This function returns the path to the FPGA Device Constraint Files folder for example `c:\program files\Altium Designer\library\FPGA\DeviceConstraintFiles`.
See also
Special Folder Paths

SpecialFolder_AltiumDeviceImages

Declaration
```
Function SpecialFolder_AltiumDeviceImages : TDynamicString;
```
Description
This function returns the path to the Altium Device Images folder for example `c:\program files\Altium Designer\library\deviceimages\`.
See also
Special Folder Paths

SpecialFolder_ApplicationData

Declaration
```
Function SpecialFolder_ApplicationData : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and settings\UserName\Application Data` folder.
See also
Special Folder Paths

SpecialFolder_CommonAllUserApplicationData

Declaration
```
Function SpecialFolder_CommonAllUserApplicationData : TDynamicString;
```
Description
This function returns the path to the Common All User Application Data folder for example `C:\Documents and Settings\All Users\Application Data\Altium Designer\`.
See also
Special Folder Paths

SpecialFolder_CommonApplicationData

Declaration
```
Function SpecialFolder_CommonApplicationData : TDynamicString;
```
Description
This function returns the path to the Common Application data folder for example `C:\Documents and Settings\User Name\Application Data\Altium Designer\`.
See also
Special Folder Paths

SpecialFolder_CommonDocumnetTemplates

Declaration
```
Function SpecialFolder_CommonDocumnetTemplates : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\Templates` folder.
See also
Special Folder Paths

SpecialFolder_CommonLocalApplicationData

Declaration
```
Function SpecialFolder_CommonLocalApplicationData : TDynamicString;
```
Description
This function returns the path to the Common Local Application data folder for example `C:\Documents and Settings\User Name\Application Data\Altium Designer\`.
See also
Special Folder Paths

SpecialFolder_CommonProgramFiles

Declaration
```
Function SpecialFolder_CommonProgramFiles : TDynamicString;
```
Description
This function returns the path to the `C:\Program Files\Common Files` folder.
See also
Special Folder Paths

SpecialFolder_CommonStartup

Declaration
```
Function SpecialFolder_CommonStartup : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\All Users\Start Menu` folder.
See also
Special Folder Paths

SpecialFolder_CommonStartupPrograms

Declaration
```
Function SpecialFolder_CommonStartupPrograms : TDynamicString;
```
Description

This function returns the path to the `C:\Documents and Settings\All Users\Start Menu\Programs` folder.

See also

Special Folder Paths

SpecialFolder_CommonFavorites

Declaration

```
Function SpecialFolder_CommonFavorites : TDynamicString;
```

Description

This function returns the path to the `C:\Documents and Settings\All Users\Favorites` folder.

See also

Special Folder Paths

SpecialFolder_ControlPanel

Declaration

```
Function SpecialFolder_ControlPanel : TDynamicString;
```

Description

This function returns the path to the Control Panel folder.

See also

Special Folder Paths

SpecialFolder_DesignExamples

Declaration

```
Function SpecialFolder_DesignExamples : TDynamicString;
```

Description

This function returns the path to the Design Examples folder. Example `C:\Program Files\Altium\Examples\`

See also

Special Folder Paths

SpecialFolder_DesignTemplates

Declaration

```
Function SpecialFolder_DesignTemplates : TDynamicString;
```

Description

This function returns the path to the DesignTemplates folder. Example `C:\Program Files\Altium\Templates\`

See also

Special Folder Paths

SpecialFolder_Desktop

Declaration

```
Function SpecialFolder_Desktop : TDynamicString;
```

Description

This function returns the path to the `C:\Documents and Settings\UserName\Desktop` folder.

See also

Special Folder Paths

SpecialFolder_DesktopLocation

Declaration
```
Function SpecialFolder_DesktopLocation : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\Desktop` folder.
See also
Special Folder Paths

SpecialFolder_Favorites

Declaration
```
Function SpecialFolder_Favorites : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\Cookies` folder.
See also
Special Folder Paths

SpecialFolder_Fonts

Declaration
```
Function SpecialFolder_Fonts : TDynamicString;
```
Description
This function returns the path to the folder where fonts are stored. For example, `C:\WinNT\Fonts`
See also
Special Folder Paths

SpecialFolder_InstalledPrinters

Declaration
```
Function SpecialFolder_InstalledPrinters : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\PrintHood` folder.
See also
Special Folder Paths

SpecialFolder_Internet

Declaration
```
Function SpecialFolder_Internet : TDynamicString;
```
Description
This function returns the path to the folder where the internet browser software is located in.
See also
Special Folder Paths

SpecialFolder_InternetCookies

Declaration
```
Function SpecialFolder_InternetCookies : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\Cookies` folder.
See also
Special Folder Paths

SpecialFolder_InternetHistory

Declaration
```
Function SpecialFolder_InternetHistory : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\Local Settings\History` folder.
See also
Special Folder Paths

SpecialFolder_InternetTemporaryFiles

Declaration
```
Function SpecialFolder_InternetTemporaryFiles : TDynamicString;
```
Description
This function returns the path to the `C:\Documents and Settings\UserName\Local Settings\Temporary Internet Files` folder.
See also
Special Folder Paths

SpecialFolder_LocalApplicationdata

Declaration
```
Function SpecialFolder_LocalApplicationData : TDynamicString;
```
Description
This function returns the path to the C:\Documents and settings\UserName\Local Settings\Application Data folder
See also
Special Folder Paths

SpecialFolder_MyComputer

Declaration
```
Function SpecialFolder_MyComputer : TDynamicString;
```
Description
This function returns the path to the MyComputer folder.
See also
Special Folder Paths

SpecialFolder_MyDesigns

Declaration
```
Function SpecialFolder_MyDesigns : TDynamicString;
```
Description
This function returns the path to the MyDesigns folder. Example C:\Documents and Settings\UserName\My Documents\My Designs
See also
Special Folder Paths

SpecialFolder_MyDocuments

Declaration
```
Function SpecialFolder_MyDocuments : TDynamicString;
```

Description
This function returns the path to the C:\Documents and Settings\UserName\Local Settings\My Documents folder.
See also
Special Folder Paths

SpecialFolder_MyMusic

Declaration
```
Function SpecialFolder_MyMusic : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\Local Settings\My Music folder.
See also
Special Folder Paths

SpecialFolder_MyNetworkPlaces

Declaration
```
Function SpecialFolder_MyNetworkPlaces : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\NetHood folder.
See also
Special Folder Paths

SpecialFolder_MyPictures

Declaration
```
Function SpecialFolder_MyPictures : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\Local Settings\My Pictures folder.
See also
Special Folder Paths

SpecialFolder_NetWorkRoot

Declaration
```
Function SpecialFolder_NetworkRoot : TDynamicString;
```
Description
This function returns the path to the Network Root directory.
See also
Special Folder Paths

SpecialFolder_NonlocalizedStartupPrograms

Declaration
```
Function SpecialFolder_NonLocalizedStartupPrograms : TDynamicString;
```
Description
This function returns the path to the Non Localized Startup Programs folder.
See also
Special Folder Paths

SpecialFolder_Printers

Declaration
```
Function SpecialFolder_Printers : TDynamicString;
```
Description
This function returns the path to the Printers folder.
See also
Special Folder Paths

SpecialFolder_Profile

Declaration
```
Function SpecialFolder_Profile : TDynamicString;
```
Description
This function returns the path to the C:\Program Files\UserName.
See also
Special Folder Paths

SpecialFolder_Programs

Declaration
```
Function SpecialFolder_Programs : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\Start Menu\Programs folder.
See also
Special Folder Paths

SpecialFolder_ProgramFiles

Declaration
```
Function SpecialFolder_ProgramFiles : TDynamicString;
```
Description
This function returns the path to the C:\Program Files folder
See also
Special Folder Paths

SpecialFolder_Recent

Declaration
```
Function SpecialFolder_Recent : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\Recent folder.
See also
Special Folder Paths

SpecialFolder_Recovery

Declaration
```
Function SpecialFolder_Recovery : TDynamicString;
```
Description
This function returns the path to the Altium Recover folder. Example C:\Documents and Settings\UserName\Application Data\Recovery\

See also
Special Folder Paths

SpecialFolder_RecycleBin

Declaration
```
Function SpecialFolder_RecycleBin : TDynamicString;
```
Description
This function returns the path to the Recycle Bin.
See also
Special Folder Paths

SpecialFolder_SendTo

Declaration
```
Function SpecialFolder_SendTo : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\SendTo folder.
See also
Special Folder Paths

SpecialFolder_StartMenuItems

Declaration
```
Function SpecialFolder_StartMenuItems : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\Recent folder.
See also
Special Folder Paths

SpecialFolder_SystemFolder

Declaration
```
Function SpecialFolder_SystemFolder : TDynamicString;
```
Description
This function returns the path to the C:\WINNT\System32 folder.
See also
Special Folder Paths

SpecialFolder_TemplatesForAllUsers

Declaration
```
Function SpecialFolder_TemplatesForAllUsers : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\All Users\Templates folder.
See also
Special Folder Paths

SpecialFolder_Temporary

Declaration
```
Function SpecialFolder_Temporary : TDynamicString;
```
Description

This function returns the path to the C:\Documents and settings\UserName\Local Settings\Temp\ folder.
See also
Special Folder Paths

SpecialFolder_TemporarySlash

Declaration
```
Function SpecialFolder_TemporarySlash : TDynamicString;
```
Description
This function returns the path to the C:\Documents and settings\UserName\Local Settings\Temp\ folder.
See also
Special Folder Paths

SpecialFolder_UserStartMenuItems

Declaration
```
Function SpecialFolder_UserStartMenuItems : TDynamicString;
```
Description
This function returns the path to the C:\Documents and Settings\UserName\Recent folder.
See also
Special Folder Paths

SpecialFolder_WindowsFolder

Declaration
```
Function SpecialFolder_WindowsFolder : TDynamicString;
```
Description
This function returns the path to the C:\WINNT folder.
See also
Special Folder Paths

String Routines

Center

Declaration
```
Function Center(Const S : TDynamicString; Width : Integer) : TDynamicString;
```
Description
Return a string centered in a blank string of specified width.
See also
String Manipulation Routines

CenterCH

Declaration
```
Function CenterCh  (Const S : TDynamicString; Ch : Char; Width : Integer) : TDynamicString;
```
Description
Returns a string centered in a string of character Ch, with specified width.
See also

String Manipulation Routines

CharStr

Declaration
```
Function CharStr   (Ch : Char; Len : Integer) : TDynamicString;
```
Description
Returns a string of length len filled with Ch
See also
String Manipulation Routines

CropStringToLength

Declaration
```
Function  CropStringToLength        (Const StringToCrop : TDynamicString;
Const MaximumLength : Integer) : TDynamicString;
```
Description
The CropStringToLength function removes leading and trailing spaces and control characters from the given string StringToCrop. The MaximumLength parameter specifies the string from index 0 to MaximumLength that will be returned by the function. The remaining portion of the string is chopped.
See also
String Manipulation Routines

GeneralStringInc

Declaration
```
Procedure GeneralStringInc (Var S : TString; Const IncValue :
TDynamicString);
```
Description
The GeneralStringInc procedure analyses the S parameter to determine if it has a number value embedded. If there is a number in the string then it increments the existing number value by one..
Example

```
S := 'Part1';
GeneralStringInc(S,'4');
//Part5
```

See also
String Manipulation Routines

GetStringFromBoolean

Declaration
```
Function  GetStringFromBoolean      (B : Boolean ) : TDynamicString;
```
Description
The GetStringFromBoolean function returns a ‘True’ if the B parameter is true otherwise a ‘False’ is returned.
See also
String Manipulation Routines

GetStringFromInteger

Declaration
```
Function  GetStringFromInteger (N : Integer) : TDynamicString;
```
Description
The GetStringFromInteger function converts any integer type to a string.
See also
String Manipulation Routines

IndentString

Declaration
```
Function  IndentString(Indent : Integer) : TDynamicString;
```
Description
The function returns you a string which specifies the amount of indentation as white spaces (#32) in this string. So an indent of 4 produces a string of four white spaces for example.
See also
String Manipulation Routines

LeftJust

Declaration
```
Function  LeftJust(Const S : TDynamicString; Width : Integer) :
TDynamicString;
```
Description
The LeftJust function left justifies a string by padding the string with (Width - Length of String) white spaces to the right of this string.
Example

```
S := LeftJust('smith',9) + '.';
//s := ‘smith    .’ (four empty spaces between the word ‘smith’ and the
fullstop ’.’)
```

See also
String Routines

PadLeft

Declaration
```
Function PadLeft(S : TDynamicString; Len : Integer) : TDynamicString;
```
Description
Returns a string left-padded to length len with blanks.
See also
String Manipulation Routines

PadLeftCh

Declaration
```
Function PadLeftCh (S : TDynamicString; Ch : Char; Len : Integer) :
TDynamicString;
```
Description
Returns a string left-padded to length len with the specified character, Ch.

See also
String Manipulation Routines

PadRight

Declaration
```
Function PadRight(S : TDynamicString; Len : Integer) : TDynamicString;
```
Description
Returns a string right-padded to length len with blanks.
See also
String Manipulation Routines

PadRightCh

Declaration
```
Function PadRightCh(S : TDynamicString; Ch : Char; Len : Integer) :
TDynamicString;
```
Description
Returns a string right-padded to length specified by the `len` parameter and with `Ch` characters.
See also
String Manipulation Routines

SameString

Declaration
Function SameString (Const S1,S2 : TDynamicString; CaseSensitive : Boolean) : Boolean;
Description
This `SameString` function compares two strings and depending on the `CaseSensitive` parameter returns a boolean result. If `CaseSensitive` is set to false, then the two strings, 'aaa' and 'AaA' are considered the same.
See also
String Manipulation Routines

StringsEqual

Declaration
```
Function StringsEqual(S1,S2 : TDynamicString) :Boolean;
```
Description
This `SameString` function compares two strings and checks whether Strings `S1` and `S2` have equal lengths and have the same contents.
See also
String Manipulation Routines

StringReplace

Syntax
```
Function StringReplace(const S, OldPattern, NewPattern: string; Flags:
TReplaceFlags): string;
```
Description
Basically this function returns a string with occurrences of one substring replaced by another substring. The `StringReplace` replaces occurrences of the substring specified by `OldPattern` with the substring specified by `NewPattern`.
Parameters

S is the source string, whose substrings are changed.
`OldPattern` is the substring to locate and replace with `NewPattern`.
`NewPattern` is the substring to substitute for occurrences of `OldPattern`.
`Flags` is a set of flags that govern how `StringReplace` locates and replaces occurrences of `OldPattern`. If `Flags` does not include `rfReplaceAll`, `StringReplace` only replaces the first occurrence of OldPattern in S. Otherwise, `StringReplace` replaces all instances of `OldPattern` with `NewPattern`. If the `Flags` parameter includes `rfIgnoreCase`, the comparison operation is case insensitive.

Notes

```
Type
  TReplaceFlags = set of (rfReplaceAll, rfIgnoreCase);
```

Example

```
Result := StringReplace(AKeys, ADelimiter, cDatabase_KeyFieldDelimiter,
[rfReplaceAll]);
```

See also

String Manipulation routines

StrToInt

Declaration

```
Function StrToInt(const S: string): Integer;
```

Description

The `StrToInt` function converts the string S, which represents an integer-type number in either decimal or hexadecimal notation, into a number.

See also

String Manipulation Routines

TrimLead

Declaration

```
Function TrimLead  (Const S : TDynamicString) : TDynamicString;
```

Description

Returns a string with leading white space removed.

See also

String Manipulation Routines

TrimTrail

Declaration

```
Function TrimTrail (Const S : TDynamicString) : TDynamicString;
```

Description

Returns a string with trailing white space removed.

See also

String Manipulation Routines

Time and Date Routines

DateString

Declaration

```
Function DateString (Const DateRecord   : TDate) : TDynamicString;
```

Description
The DateString function returns a TString representing a date in ‘12-Jan-1985’ format.
See also
Time and Date Routines

GetCurrentDate

Declaration
```
Procedure GetCurrentDate     (Var   DateRecord   : TDate);
```
Description
The `GetCurrentDate` procedure is based on the Window API’s `DecodeDate` procedure which breaks the value specified as the `Date` parameter into Year, Month, and Day values. If the given `TDateTime` value is less than or equal to zero, the year, month, and day return parameters are all set to zero.
See also
Time and Date Routines

GetCurrentDateString

Declaration
```
Function  GetCurrentDateString : TDynamicString;
```
Description
The `GetCurrentDateString` function returns a `TString` representing date in ‘12-Jan-1985’ format
See also
Time and Date Routines

GetCurrentTimeString

Declaration
```
Function GetCurrentTimeString : TDynamicString;
```
Description
The `GetCurrentTimeString` function returns a TString representing a time of day in HH:MM:SS format.
See also
Time and Date Routines

GetCurrentTimeRec

Declaration
```
Procedure GetCurrentTimeRec (Var TimeRecord : TTime);
```
Description
The `GetCurrentTimeRec` procedure is based on WinAPI’s `DecodeTime` function which breaks the `TDateTime` record into hours, minutes, seconds, and milliseconds.
See also
Time and Date Routines

GetDateAndTimeStamp

Declaration
```
Function  GetDateAndTimeStamp : TDynamicString;
```
Description
This function returns the string containing the current date and the time.
See also
Time and Date Routines

GetElapsedTime

Declaration
```
Procedure GetElapsedTime (Const Start : TTime; Const Stop : TTime;Var Elapsed
: TTime);
```
Description
The GetElapsedTime procedure returns the Elapsed value in seconds between the Start and Stop timing intervals.
See also
Time and Date Routines

GetElapsedTimeDate

Declaration
```
Procedure GetElapsedTimeDate (Const Start     : TTime;
                              Const Stop      : TTime;
                              Var   Elapsed   : TTime;
                              Const StartDate : TDate;
                              Const StopDate  : TDate);
```
Description
The GetElapsedTimeDate procedure returns the Elapsed value derived from the StartDate, StopDate dates and Start, Stop times. The results can be retrieved as a string by the TimeString_Elapsed function.
See also
Time and Date Routines

GetFileDateString

Declaration
```
Function  GetFileDateString(Const AFileName : TDynamicString) :
TDynamicString;
```
Description
The GetCurrentDateString function returns a String representing date in ‘12-Jan-1985’ format for example.
See also
Time and Date Routines

GetMilliSecondTime

Declaration
```
Function GetMilliSecondTime : Integer;
```
Description
The GetMilliSecondTime function retrieves the number of milliseconds that have elapsed since Windows was started.
See also
Time and Date Routines

MakeDateAndTimeStampedFileName

Declaration
```
Function  MakeDateAndTimeStampedFileName(BaseName : TDynamicString) :
TDynamicString;
```

Description
This function returns the date and time inserted in the base file name string.
See also
Time and Date Routines

SecondsToTimeRecord

Declaration
```
Procedure SecondsToTimeRecord(Var TimeRecord : TTime; Const Seconds :
Integer);
```
Description
This procedure does the reverse of the `TimeRecordToSeconds` procedure. It converts the seconds information into the `TTime` structure type.
See also
Time and Date Routines

TimeString_elapsed

Declaration
```
Function  TimeString_Elapsed (Const TimeRecord   : TTime) : TDynamicString;
```
Description
This function returns the string containing the Time information that has elapsed. To find the timing information, invoke the `GetElapsedTimeDate` or `GetElapsedTime` function.
Example

```
Var
   ElapsedTime : TTime;
Begin
   GetCurrentTimeRec (EndTime);
   GetCurrentDate (EndDate);
   GetElapsedTimeDate (StartTime, EndTime, ElapsedTime, StartDate, EndDate);
   ShowInfo('Time Elapsed : ' + TimeString_Elapsed(ElapsedTime));
End;
```

See also
Time and Date Routines

TimeString

Declaration
```
Function  TimeString         (Const TimeRecord   : TTime) : TDynamicString;
```
Description
The TimeString function returns a TString representing a time of day in HH:MM:SS format.
See also
Time and Date Routines

TimeRecordToSeconds

Declaration
```
Procedure TimeRecordToSeconds(Const TimeRecord   : TTime;  Var   Seconds
: Integer);
```
Description

This procedure converts a TTime type structure into number of seconds. This procedure is used for GetElapsedTime and GetElapsedTimeDate procedures.

See also

Time and Date Routines

WaitMilliSecondDelay

Declaration

```
Procedure WaitMilliSecondDelay(N : Integer);
```

Description

The `WaitMilliSecondDelay` function provides a delay in the code in milli-seconds as specified by the `N` integer value. This is useful if a function in the software needs delaying for a while before doing something else giving the software a chance to catch up. This function uses the `GetMilliSecondTime` function.

Example

```
WaitMilliSecondDelay(1000); // waits for 1 second. 1000 milliseconds = 1 second.
```

See also

Time and Date Routines

Functions from ClientProcs unit

```
Function  ClientAPI_GetPrefAnimatedPanels
: Boolean;
Function  ClientAPI_GetPrefSaveToolsLayout
: Boolean;
Function  ClientAPI_GetPrefAutoTransparency
: Boolean;
Function  ClientAPI_GetPrefDynamicAutoTransparency
: Boolean;
Function  ClientAPI_GetPrefSuppressStartupScreen
: Boolean;
Function  ClientAPI_GetPrefTransparencyHighest
: Integer;
Function  ClientAPI_GetPrefTransparencyLowest
: Integer;
Function  ClientAPI_GetPrefTransparencyForce
: Integer;
Function  ClientAPI_GetPrefPopupPanelDelay
: Integer;
Function  ClientAPI_GetPrefHidePanelDelay
: Integer;
Function  ClientAPI_GetPrefAnimatedPanelSpeed
: Integer;
Function  ClientAPI_GetPrefPathInTitleBar
: Boolean;
Function  ClientAPI_GetPrefUseShadow
: Boolean;
Function  ClientAPI_GetPrefUseLuna
: Boolean;
Function  ClientAPI_GetPrefHideFloatingPanels
```

```
: Boolean;
Function  ClientAPI_GetPrefRestoreOpenDocuments
        : Boolean;
Function  ClientAPI_GetPrefOpenTasksIfNothingOpen
: Boolean;
Function  ClientAPI_GetPrefHideBinderViewTabs
: Boolean;
Function  ClientAPI_GetPrefNoRestoreKindCount
               : Integer;
Procedure ClientAPI_GetPrefNoRestoreKind                        (Index
: Integer; Buffer : PChar);

Procedure ClientAPI_SetPrefAnimatedPanels                       (Value
: Boolean);
Procedure ClientAPI_SetPrefSaveToolsLayout                      (Value
: Boolean);
Procedure ClientAPI_SetPrefAutoTransparency                     (Value
: Boolean);
Procedure ClientAPI_SetPrefDynamicAutoTransparency              (Value
: Boolean);
Procedure ClientAPI_SetPrefSuppressStartupScreen                (Value
: Boolean);
Procedure ClientAPI_SetPrefTransparencyHighest                  (Value
: Integer);
Procedure ClientAPI_SetPrefTransparencyLowest                   (Value
: Integer);
Procedure ClientAPI_SetPrefTransparencyForce                    (Value
: Integer);
Procedure ClientAPI_SetPrefPopupPanelDelay                      (Value
: Integer);
Procedure ClientAPI_SetPrefHidePanelDelay                       (Value
: Integer);
Procedure ClientAPI_SetPrefAnimatedPanelSpeed                   (Value
: Integer);
Procedure ClientAPI_SetPrefPathInTitleBar                       (Value
: Boolean);
Procedure ClientAPI_SetPrefUseShadow                            (Value
: Boolean);
Procedure ClientAPI_SetPrefUseLuna                              (Value
: Boolean);
Procedure ClientAPI_SetPrefHideFloatingPanels                   (Value
: Boolean);
Procedure ClientAPI_SetPrefRestoreOpenDocuments                 (Value
: Boolean);
Procedure ClientAPI_SetPrefOpenTasksIfNothingOpen               (Value
: Boolean);
Procedure ClientAPI_SetPrefHideBinderViewTabs                   (Value
: Boolean);
Procedure ClientAPI_SetPrefNoRestoreKindClear;
Procedure ClientAPI_SetPrefNoRestoreKindAdd                     (Value
: PChar);
```

```
Function  ClientAPI_GetPrefRememberFormForDocKind
: Boolean;
Procedure ClientAPI_SetPrefRememberFormForDocKind             (Value
: Boolean);
Procedure ClientAPI_SetAutoShowComponentSymbols               (Value
: Boolean);
Function  ClientAPI_GetAutoShowComponentSymbols
: Boolean;


Procedure ClientAPI_ShowProductStartup (Bitmap       : TDynamicString);
Procedure ClientAPI_HideProductStartup;
Procedure ClientAPI_AddStartupMessage  (S            : TDynamicString);
Procedure ClientAPI_AddShutdownMessage (S            : TDynamicString);

Procedure ClientAPI_Synchronize (Const ASync : IThreadSynchronize);
Procedure ClientAPI_CheckSynchronize;

Function ClientAPI_GetCurrentOutputGenerator : IUnknown;
Procedure ClientAPI_SetCurrentOutputGenerator(Const Generator : IUnknown);

Function  ClientAPI_GetBuiltInNavigationBar          : Boolean;
Procedure ClientAPI_SetBuiltInNavigationBar   (Value : Boolean);
Function  ClientAPI_GetAlwaysShowNavBarInTasks       : Boolean;
Procedure ClientAPI_SetAlwaysShowNavBarInTasks(Value : Boolean);
{.....................................................................
.............}
{.....................................................................
.............}
Function  ClientAPI_GetFavoritesThumbnailSize       : TSize;
Procedure ClientAPI_SetFavoritesThumbnailSize(Value : TSize);
{.....................................................................
.............}
{.....................................................................
.............}
Function  ClientAPI_GetGroupingInDocumentsBar          :
TDocumentsBarGrouping;
Procedure ClientAPI_SetGroupingInDocumentsBar    (Value :
TDocumentsBarGrouping);
Function  ClientAPI_GetEqualButtonsInDocumentsBar      : Boolean;
Procedure ClientAPI_SetEqualButtonsInDocumentsBar(Value : Boolean);
Function  ClientAPI_GetAutoHideDocumentsBar            : Boolean;
Procedure ClientAPI_SetAutoHideDocumentsBar      (Value : Boolean);
Function  ClientAPI_GetMultilineDocumentsBar           : Boolean;
Procedure ClientAPI_SetMultilineDocumentsBar     (Value : Boolean);
Function  ClientAPI_GetMiddleClickClosesDocumentTab      : Boolean;
Procedure ClientAPI_SetMiddleClickClosesDocumentTab(Value : Boolean);
Function  ClientAPI_GetIntegratedHelpSystem            : Boolean;
Procedure ClientAPI_SetIntegratedHelpSystem      (Value : Boolean);
Function  ClientAPI_GetUseSystemLocaleLanguage         : Boolean;
Procedure ClientAPI_SetUseSystemLocaleLanguage   (Value : Boolean);
```

```
Function  ClientAPI_GetUseLocalizedDialogs             : Boolean;
Procedure ClientAPI_SetUseLocalizedDialogs       (Value : Boolean);
Function  ClientAPI_GetUseLocalizedResources           : Boolean;
Procedure ClientAPI_SetUseLocalizedResources     (Value : Boolean);
Function  ClientAPI_GetVSStyleCtrlTab                  : Boolean;
Procedure ClientAPI_SetVSStyleCtrlTab            (Value : Boolean);
Function  ClientAPI_GetActivateLastActiveOnClose       : Boolean;
Procedure ClientAPI_SetActivateLastActiveOnClose (Value : Boolean);
{.....................................................................
.................}

Function ClientAPI_GetHelpFileAndTopic(Const AHelpTopicID : WideString; Out
HelpFileName, HelpTopicName : WideString) : Boolean;

Function  ClientAPI_UpdateFont(Var Font : TLogFont) : LongBool;
Procedure ClientAPI_SetErrorInfo(Const ErrorMsg, ErrorReport : WideString;
ErrorAddr : Pointer);
Procedure ClientAPI_ClearErrorInfo;
Procedure ClientAPI_HandleException(Const Message : WideString);

Procedure ClientAPI_QueryUpdatesInfo        (Var   UpdatesURL,
UpdatesNetworkPath : WideString; Var UpdatesUseNetworkPath : LongBool; Var
UpdatesPathToDownloadUpdates : WideString;
Var CheckFrequency : TWebUpdate_CheckFrequency); Stdcall;

Procedure ClientAPI_SetUpdatesInfo          (Const UpdatesURL,
UpdatesNetworkPath : WideString;     UpdatesUseNetworkPath : LongBool; Const
UpdatesPathToDownloadUpdates : WideString;
        CheckFrequency : TWebUpdate_CheckFrequency); Stdcall;
```
