# PB.Ex FTP

## Extended FTP Library for PureBasic

***by RSBasic***

*PureBasic can neither SFTP (SSH File Transfer Protocol) nor FTPS (FTP over SSL/TLS).*

*With this library this is possible.*

*The functions are built in the same style as the FTP library of PureBasic and FTP is also supported.*

# Table of Contents

# 1 PB.Ex FTP

PureBasic can neither SFTP (SSH File Transfer Protocol) nor FTPS (FTP over SSL/TLS).
With this library this is possible. The functions are built in the same style as the FTP library of PureBasic and FTP is also supported.

System requirements:
- Windows XP or higher
- .NET Framework 4 or higher
- Unicode activation (standard from PB 5.50)

License: This DLL file is free of charge and may be used privately and commercially without naming a name and without a link to the homepage.

Download: http://www.rsbasic.de/downloads/download.php?file=PB.Ex_FTP.zip

I would be very pleased about feedbacks, improvement suggestions, error messages or wishes.
Thanks:)

RSBasic
Website: http://www.rsbasic.de

## 1.1 OpenFTPEx

Syntax:
Result = OpenFTPEx(ID, Protocol, ServerName$, Port, User$, Password$, Charset, @ErrorOutput)

Description: Establishes a connection to the server.

Parameter:
- ID: A unique number for the connection. PB_Any can be used to generate the number automatically.
- Protocol: Sets the protocol (FTP, SFTP, FTPS) for the connection: #PBEx_FTP_Protocol_FTP, #PBEx_FTP_Protocol_SFTP, #PBEx_FTP_Protocol_FTPS_Implicit, #PBEx_FTP_Protocol_FTPS_Explicit
- ServerName$: The domain or IP address of the server.
- Port: The port number for the connection.
- User$: The user name for logging in.
- Password$: The password for logging in.
- Charset: Specifies the character set: #PB_UTF8, #PB_Ascii, #PB_Unicode - SFTP uses UTF-8 by default.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful. If #PB_Any is used, the ID is returned.

## 1.2 CloseFTPEx

Syntax:
Result = CloseFTPEx(ID, @ErrorOutput)

Description: Closes the open connection to the server.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.3 CheckFTPConnectionEx

Syntax:
Result = CheckFTPConnectionEx(ID, @ErrorOutput)

Description: Checks if the connection to the server still exists.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The connection to the server exists.

## 1.4 IsFTPEx

Syntax:
Result = IsFTPEx(ID, @ErrorOutput)

Description: Checks whether the ID was initialized correctly.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The ID is valid.

## 1.5 ExamineFTPDirectoryEx

Syntax:
Result = ExamineFTPDirectoryEx(ID, @ErrorOutput)

Description: Starts the list from the current directory.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.6 FinishFTPDirectoryEx

Syntax:
Result = FinishFTPDirectoryEx(ID, @ErrorOutput)

Description: Closes the list of the current directory.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.7 NextFTPDirectoryEntryEx

Syntax:
Result = NextFTPDirectoryEntryEx(ID, @ErrorOutput)

Description: The next folder or file is determined.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: Another folder or file exists.

## 1.8 FTPDirectoryEntryNameEx

Syntax:
Result = FTPDirectoryEntryNameEx(ID, @Output, @ErrorOutput)

Description: The folder or file name is determined.

Parameter:
- ID: The number of the connection.
- @Output: The folder or file name is stored in the string variable.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.9 FTPDirectoryEntrySizeEx

Syntax:
Result = FTPDirectoryEntrySizeEx(ID, @ErrorOutput)

Description: The size of the file is determined.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
The size of the file is returned.

## 1.10 FTPDirectoryEntryTypeEx

Syntax:
Result = FTPDirectoryEntryTypeEx(ID, @ErrorOutput)

Description: Checks whether the entry is a file or a folder.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: It's a file. #PB_FTP_File can be used.
2: It's a folder. PB_FTP_Directory can be used.

## 1.11 FTPDirectoryEntryDateEx

Syntax:
Result = FTPDirectoryEntryDateEx(ID, @ErrorOutput)

Description: Determines the processing date of the file or folder.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
The processing date of the file or folder. The value can be used with the Date library.

## 1.12 FTPDirectoryEntryAttributesEx

Syntax:
Result = FTPDirectoryEntryAttributesEx(ID, @ErrorOutput)

Description: Determines the specified attributes of the file or folder.

Parameter:
- ID: The number of the connection.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
The specified attributes of the file or folder are returned and can be used.
#PB_FTP_ReadUser
#PB_FTP_WriteUser
#PB_FTP_ExecuteUser
#PB_FTP_ReadGroup
#PB_FTP_WriteGroup
#PB_FTP_ExecuteGroup
#PB_FTP_ReadAll
#PB_FTP_WriteAll
#PB_FTP_ExecuteAll

## 1.13 GetFTPDirectoryEx

Syntax:
Result = GetFTPDirectoryEx(ID, @Output, @ErrorOutput)

Description: Determines the current path.

Parameter:
- ID: The number of the connection.
- @Output: The current path is stored in the string variable.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.14 SetFTPDirectoryEx

Syntax:
Result = SetFTPDirectoryEx(ID, DirectoryName$, @ErrorOutput)

Description: Opens a subfolder.

Parameter:
- ID: The number of the connection.
- DirectoryName$: Name of the folder to open.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.15 CreateFTPDirectoryEx

Syntax:
Result = CreateFTPDirectoryEx(ID, DirectoryName$, @ErrorOutput)

Description: Creates a new directory.

Parameter:
- ID: The number of the connection.
- DirectoryName$: Name of the folder to be created.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.16 DeleteFTPDirectoryEx

Syntax:
Result = DeleteFTPDirectoryEx(ID, DirectoryName$, @ErrorOutput)

Description: Deletes a directory.

Parameter:
- ID: The number of the connection.
- DirectoryName$: Name of the folder you want to delete.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.17 DeleteFTPFileEx

Syntax:
Result = DeleteFTPFileEx(ID, FileName$, @ErrorOutput)

Description: Deletes a file.

Parameter:

- ID: The number of the connection.
- FileName$: Name of the file to be deleted.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.18 RenameFTPFileEx

Syntax:
Result = RenameFTPFileEx(ID, FileName$, NewFileName$, @ErrorOutput)

Description: Renames a file.

Parameter:
- ID: The number of the connection.
- FileName$: Name of the file to be renamed.
- NewFileName$: The new name of the file.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.19 ReceiveFTPFileEx

Syntax:
Result = ReceiveFTPFileEx(ID, RemoteFileName$, FileName$, @ErrorOutput)

Description: Downloads a file.

Parameter:
- ID: The number of the connection.
- RemoteFileName$: Name of the file to download.
- FileName$: Local destination path.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

## 1.20 SendFTPFileEx

Syntax:
Result = SendFTPFileEx(ID, FileName$, RemoteFileName$, @ErrorOutput)

Description: Uploads a file.

Parameter:
- ID: The number of the connection.
- FileName$: Local path of the file to upload.
- RemoteFileName$: Name of the file.
- @ErrorOutput: If an error occurs, the error message is stored in the string variable.

Return value:
1: The process was successful.

# 2 Example Code

```
EnableExplicit

Define PBEx_FTP

#PBEx_FTP_Protocol_FTP = 1
#PBEx_FTP_Protocol_SFTP = 2
#PBEx_FTP_Protocol_FTPS_Implicit = 3
#PBEx_FTP_Protocol_FTPS_Explicit = 4

CompilerIf #PB_Compiler_Processor = #PB_Processor_x86
  PBEx_FTP = OpenLibrary(#PB_Any, "PB.Ex_FTP_x86.dll")
CompilerElseIf #PB_Compiler_Processor = #PB_Processor_x64
  PBEx_FTP = OpenLibrary(#PB_Any, "PB.Ex_FTP_x64.dll")
CompilerEndIf

If PBEx_FTP
  Prototype OpenFTPEx(ID, Protocol, ServerName.p-Unicode, Port, User.p-Unicode,
Password.p-Unicode, Charset, ErrorOutput)
  Define OpenFTPEx.OpenFTPEx = GetFunction(PBEx_FTP, "OpenFTPEx")
  Prototype CloseFTPEx(ID, ErrorOutput)
  Define CloseFTPEx.CloseFTPEx = GetFunction(PBEx_FTP, "CloseFTPEx")
  Prototype CheckFTPConnectionEx(ID, ErrorOutput)
  Define CheckFTPConnectionEx.CheckFTPConnectionEx = GetFunction(PBEx_FTP,
"CheckFTPConnectionEx")
  Prototype IsFTPEx(ID, ErrorOutput)
  Define IsFTPEx.IsFTPEx = GetFunction(PBEx_FTP, "IsFTPEx")
  Prototype ExamineFTPDirectoryEx(ID, ErrorOutput)
  Define ExamineFTPDirectoryEx.ExamineFTPDirectoryEx = GetFunction(PBEx_FTP,
"ExamineFTPDirectoryEx")
  Prototype FinishFTPDirectoryEx(ID, ErrorOutput)
  Define FinishFTPDirectoryEx.FinishFTPDirectoryEx = GetFunction(PBEx_FTP,
"FinishFTPDirectoryEx")
  Prototype NextFTPDirectoryEntryEx(ID, ErrorOutput)
  Define NextFTPDirectoryEntryEx.NextFTPDirectoryEntryEx = GetFunction(PBEx_FTP,
"NextFTPDirectoryEntryEx")
  Prototype FTPDirectoryEntryNameEx(ID, Output, ErrorOutput)
  Define FTPDirectoryEntryNameEx.FTPDirectoryEntryNameEx = GetFunction(PBEx_FTP,
"FTPDirectoryEntryNameEx")
  Prototype FTPDirectoryEntrySizeEx(ID, ErrorOutput)
  Define FTPDirectoryEntrySizeEx.FTPDirectoryEntrySizeEx = GetFunction(PBEx_FTP,
"FTPDirectoryEntrySizeEx")
  Prototype FTPDirectoryEntryTypeEx(ID, ErrorOutput)
  Define FTPDirectoryEntryTypeEx.FTPDirectoryEntryTypeEx = GetFunction(PBEx_FTP,
"FTPDirectoryEntryTypeEx")
  Prototype FTPDirectoryEntryDateEx(ID, ErrorOutput)
  Define FTPDirectoryEntryDateEx.FTPDirectoryEntryDateEx = GetFunction(PBEx_FTP,
"FTPDirectoryEntryDateEx")
  Prototype FTPDirectoryEntryAttributesEx(ID, ErrorOutput)
  Define FTPDirectoryEntryAttributesEx.FTPDirectoryEntryAttributesEx =
GetFunction(PBEx_FTP, "FTPDirectoryEntryAttributesEx")
  Prototype GetFTPDirectoryEx(ID, Output, ErrorOutput)
  Define GetFTPDirectoryEx.GetFTPDirectoryEx = GetFunction(PBEx_FTP,
"GetFTPDirectoryEx")
  Prototype SetFTPDirectoryEx(ID, DirectoryName.p-Unicode, ErrorOutput)
  Define SetFTPDirectoryEx.SetFTPDirectoryEx = GetFunction(PBEx_FTP,
"SetFTPDirectoryEx")
  Prototype CreateFTPDirectoryEx(ID, DirectoryName.p-Unicode, ErrorOutput)
  Define CreateFTPDirectoryEx.CreateFTPDirectoryEx = GetFunction(PBEx_FTP,
"CreateFTPDirectoryEx")
  Prototype DeleteFTPDirectoryEx(ID, DirectoryName.p-Unicode, ErrorOutput)
  Define DeleteFTPDirectoryEx.DeleteFTPDirectoryEx = GetFunction(PBEx_FTP,
"DeleteFTPDirectoryEx")
  Prototype DeleteFTPFileEx(ID, FileName.p-Unicode, ErrorOutput)
```

```
  Define DeleteFTPFileEx.DeleteFTPFileEx = GetFunction(PBEx_FTP, "DeleteFTPFileEx")
  Prototype RenameFTPFileEx(ID, FileName.p-Unicode, NewFileName.p-Unicode,
ErrorOutput)
  Define RenameFTPFileEx.RenameFTPFileEx = GetFunction(PBEx_FTP, "RenameFTPFileEx")
  Prototype ReceiveFTPFileEx(ID, RemoteFileName.p-Unicode, FileName.p-Unicode,
ErrorOutput)
  Define ReceiveFTPFileEx.ReceiveFTPFileEx = GetFunction(PBEx_FTP,
"ReceiveFTPFileEx")
  Prototype SendFTPFileEx(ID, FileName.p-Unicode, RemoteFileName.p-Unicode,
ErrorOutput)
  Define SendFTPFileEx.SendFTPFileEx = GetFunction(PBEx_FTP, "SendFTPFileEx")

  Define ErrorOutput$ = Space(1024)
  Define FileName$ = Space(#MAX_PATH)

  Debug "SFTP..."

  If OpenFTPEx(1, #PBEx_FTP_Protocol_SFTP, "test.rebex.net", 22, "demo", "password",
#PB_UTF8, @ErrorOutput$)
    SetFTPDirectoryEx(1, "pub", @ErrorOutput$)
    SetFTPDirectoryEx(1, "example", @ErrorOutput$)
    If ExamineFTPDirectoryEx(1, @ErrorOutput$)
      While NextFTPDirectoryEntryEx(1, @ErrorOutput$)
        FTPDirectoryEntryNameEx(1, @FileName$, @ErrorOutput$)
        Debug FileName$
      Wend
    EndIf
    CloseFTPEx(1, @ErrorOutput$)
    Else
    Debug ErrorOutput$
  EndIf

  Debug ""
  Debug "FTP..."

  If OpenFTPEx(1, #PBEx_FTP_Protocol_FTP, "rsbasic.de", 21, "webika3rg_qch3ai",
"PureBasic!2016", #PB_UTF8, @ErrorOutput$)
    If ExamineFTPDirectoryEx(1, @ErrorOutput$)
      While NextFTPDirectoryEntryEx(1, @ErrorOutput$)
        FTPDirectoryEntryNameEx(1, @FileName$, @ErrorOutput$)
        Debug FileName$
      Wend
    EndIf
    CloseFTPEx(1, @ErrorOutput$)
    Else
    Debug ErrorOutput$
  EndIf

  Debug ""
  Debug "FTPS explicit..."

  If OpenFTPEx(1, #PBEx_FTP_Protocol_FTPS_Explicit, "test.rebex.net", 21, "demo",
"password", #PB_UTF8, @ErrorOutput$)
    If ExamineFTPDirectoryEx(1, @ErrorOutput$)
      While NextFTPDirectoryEntryEx(1, @ErrorOutput$)
        FTPDirectoryEntryNameEx(1, @FileName$, @ErrorOutput$)
        Debug FileName$
      Wend
    EndIf
    CloseFTPEx(1, @ErrorOutput$)
    Else
    Debug ErrorOutput$
  EndIf

  Debug ""
  Debug "FTPS implicit..."
```

```
  If OpenFTPEx(1, #PBEx_FTP_Protocol_FTPS_Implicit, "test.rebex.net", 990, "demo",
"password", #PB_UTF8, @ErrorOutput$)
    If ExamineFTPDirectoryEx(1, @ErrorOutput$)
      While NextFTPDirectoryEntryEx(1, @ErrorOutput$)
        FTPDirectoryEntryNameEx(1, @FileName$, @ErrorOutput$)
        Debug FileName$
      Wend
    EndIf
    CloseFTPEx(1, @ErrorOutput$)
    Else
    Debug ErrorOutput$
  EndIf

  CloseLibrary(PBEx_FTP)
EndIf
```