Object REXX for Windows

REXX FTP Library Functions (RxFtp)

Version 2.1

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix B. Notices" on page 25.

First Edition, March 2001

This edition applies to Version 2.1 of IBM® Object REXX for Windows Interpreter Edition (5639-M69) and Development Edition (5639-M68), and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

Contents

Chapter 1. What Is RxFtp?

RxFtp is a REXX File Transfer Protocol (FTP) application program interface (API) package that provides access to FTP commands from your REXX program. This documentation covers version 2.3 of the RxFtp API.

The RxFtp function names are similar to those of their corresponding FTP commands. It is assumed that you are familiar with the basic FTP. For information on the FTP commands, refer to the TCP/IP or FTP documentation provided with your operating system.

The RxFtp package requires TCP/IP support to be active on your system.

The FTP client runs on your local system and the FTP server runs on a remote system.

Chapter 2. Registration and Removal of the RxFtp API Functions

The RxFtp package is contained in the file **rxftp.dll**. This dynamic link library (DLL) must be placed in a directory listed in your PATH. To get access to the functions in the RxFtp package, execute the following REXX code:

```
if RxFuncQuery('FtpDropFuncs') then
  do
    rc = RxFuncAdd("FTPLoadFuncs","rxftp","FTPLoadFuncs")
    rc = FtpLoadFuncs()
  end
```

To unload the DLL, call the FtpDropFuncs() function and then exit all CMD.EXE shells. After exiting all command shells, the DLL is dropped by Windows and can be deleted or replaced.

Chapter 3. Return Values

Return values are either FTP errors or are set individually for particular functions.

Error Codes

Many RxFtp functions return an FTP error code. It can have one of the following values:

0 if the call was successful

–1 if an error occurred during an FTP function

The other RxFtp functions — those for which individual return values are set — are as follows:

- FtpLoadFuncs()
- FtpDropFuncs()
- FtpVersion()
- FtpSetUser()
- FtpSetBinary()
- FtpTrace()
- FtpTraceLog()
- FtpTraceLogOff()

Their return codes are given in the descriptions of the individual functions.

If the function returns a –1, the variable FTPERRNO is set to one of the following values, or any other numeric value:

FTPSERVICE
Unknown service

FTPHOST
Unknown host

FTPSOCKET
Unable to obtain socket

FTPCONNECT
Unable to connect to the server

FTPLOGIN
Login failed

FTPABORT
Transfer stopped

FTPLOCALFILE
Cannot open the local file

FTPDATACONN
Cannot initialize data connection

FTPCOMMAND
Command failed

FTPPROXYTHIRD
The proxy server does not support third-party transfers

FTPNOPRIMARY
No primary connection for the proxy transfer

Chapter 4. Functions

Most RxFtp functions are similar to their corresponding FTP commands:

- FtpLoadFuncs()
- FtpDropFuncs()
- FtpVersion()
- FtpSetUser()
- FtpSetActiveMode()
- FtpSetBinary()
- FtpLogoff()
- FtpAppend()
- FtpDelete()
- FtpRename()
- FtpGet()
- FtpPut()
- FtpPutUnique()
- FtpLs()
- FtpDir()
- FtpChDir()
- FtpMkDir()
- FtpRmDir()
- FtpPwd()
- FtpQuote()
- FtpSite()
- FtpSys()
- FtpProxy()
- FtpPing()
- FtpTrace()
- FtpTraceLog()
- FtpTraceLogOff()

FtpLoadFuncs()

The FtpLoadFuncs() call loads all functions in the RxFtp package.

Syntax:

```
rc = FtpLoadFuncs()
```

All parameters that you supply are only used to bypass copyright information.

Return values:

0 if the call was successful

–1 if an error occurred during the call

FtpDropFuncs()

The FtpDropFuncs() call drops all functions in the RxFtp package.

Syntax:

```
call FtpDropFuncs
```

FtpVersion()

The FtpVersion() call identifies the version of the RxFtp package.

Syntax:

```
call FtpVersion version
```

where:

version
is the version of the RxFtp API that you are using.

The return value can be ignored because the version is returned in the parameter *version* that you pass.

FtpSetUser()

The FtpSetUser() call sets the host name, user ID, password, and, optionally, the user's account for the remote FTP server. These parameters remain current during the entire FTP session, or until they are respecified.

To prevent unauthorized access to the remote FTP server, you can blank out the password in your REXX program when you have finished or use the

FtpLogoff() function. You can also create a REXX prompt for password specification using the SysGetKey function.

Syntax:

```
rc = FtpSetUser(host,userid,password<,account>)
```

where:

host
is the name of the remote host to which you want to connect.

userid
identifies you to the FTP server.

password
supplies a password to the remote FTP server.

account
supplies host-dependent accounting information.

Return values:

1 if the call was successful

0 if the call was *not* successful.

FtpSetActiveMode()

The FtpSetActiveMode() call puts the FTP server into either active or passive mode. The default is the passive mode because it supports transfers through firewalls. The passive mode falls back into active mode when the server does not accept a passive-mode connection.

It can happen that an FTP server does not support passive mode, making transfers impossible, as in the case of the S/390 FTP servers. In this case, you can explicitly select the active mode for the current session. The active mode is used until you reset this mode with a FtpSetActiveMode(0) call. Note that in active mode you cannot transfer files through a firewall.

Syntax:

```
rc = FtpSetActiveMode("1"|"0")
```

where:

″**1**″
puts the FTP server into active mode.

″**0**″
puts the FTP server into passive mode with fallback to active mode.

The return value is an FTP error code.

FtpSetBinary()

The FtpSetBinary() call sets the default transfer mode to binary or ASCII for functions that can use these modes. You can temporarily override the set mode by any function that takes "Binary"|"Ascii" as an optional parameter.

The value of the parameter you use for the FtpSetBinary() call can be passed as an abbreviation. For "Binary", specify "b" or "BIN". For "Ascii", specify "as".

Syntax:

```
rc = FtpSetBinary("Binary"|"Ascii")
```

where:

"**Binary**"
sets the file transfer mode to binary

"**Ascii**"
sets the file transfer mode to ASCII (flat text)

Return values:

0 if the call was successful

–1 if an error occurred during the call

FtpLogoff()

The FtpLogoff() call ends all FTP sessions with the remote FTP server. The host, user ID, password, and account are reset after this call.

The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpLogoff()
```

The return value is an FTP error code.

FtpAppend()

The FtpAppend() call copies a local file to the remote FTP server and adds it to the end of a file on the remote host. Optionally, you can specify the transfer to occur in binary mode or ASCII mode.

The remote host running the FTP server is specified with the FtpSetUser() call.

If you do not specify the transfer mode with this call, the mode specified with the FtpSetBinary() call is used.

Syntax:

```
rc = FtpAppend(localFile,remoteFile<,"Binary"|"Ascii">)
```

where:

localFile
is the name of the local file to be copied.

remoteFile
is the name of the file on the remote host to which the local file is added.

"**Binary**"
sets the file transfer mode to binary or image.

"**Ascii**"
sets the file transfer type to ASCII (flat text).

The return value is an FTP error code.

FtpDelete()

The FtpDelete() call deletes a single file on the remote host. The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
c = FtpDelete(remoteFile)
```

where:

remoteFile
is the name of the file to be deleted on the remote host.

The return value is an FTP error code.

FtpRename()

The FtpRename() call changes the name of a file on the remote host. The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpRename(oldFile,newFile)
```

where:

oldFile
is the original name of the file on the remote host.

newFile
is the new name of the file on the remote host.

The return value is an FTP error code.

FtpGet()

The FtpGet() call copies a single file from the remote FTP server to the local FTP client. The copy and the file to be copied need not have the same name. Optionally, you can specify the transfer to occur in binary or text (ASCII) mode.

The remote host running the FTP server is specified with the FtpSetUser() call.

If you do not specify the transfer mode with this call, the mode specified with the FtpSetBinary() call is used.

Syntax:

```
rc = FtpGet(localFile,remoteFile<,"Binary"|"Ascii">)
```

where:

localFile
is the name of the copy on the local host.

remoteFile
is the name of the file to be copied from the remote FTP server.

"**Binary**"
sets the file transfer mode to binary or image.

"**Ascii**"
sets the file transfer type to ASCII (flat text).

The return value is an FTP error code.

FtpPut()

The FtpPut() call copies a single file from the local FTP client to a remote FTP server. The copy and the file to be copied need not have the same name. Optionally, you can specify the transfer to occur in binary or text (ASCII) mode.

The remote host running the FTP server is specified with the FtpSetUser() call.

If you do not specify the transfer mode with this call, the mode specified with the FtpSetBinary() call is used.

Syntax:

```
rc = FtpPut(localFile,remoteFile<,"Binary"|"Ascii">)
```

where:

localFile
is the name of the file to be copied from the local FTP client.

remoteFile
is the name of the copy on the remote host.

"**Binary**"
sets the file transfer mode to binary or image.

"**Ascii**"
sets the file transfer type to ASCII (flat text).

The return value is an FTP error code.

FtpPutUnique()

The FtpPutUnique() call copies a single file from the local host to a remote host. If the name of the file is not unique on the remote host, the file is assigned a unique name.

The copy and the file to be copied must not have the same name. Optionally, you can specify the transfer to occur in binary mode or text (ASCII) mode.

The remote host running the FTP server is specified with the FtpSetUser() call.

If you do not specify the transfer mode with this call, the mode specified with the FtpSetBinary() call is used.

Syntax:

```
rc = FtpPutUnique(localFile,remoteFile<,"Binary"|"Ascii">)
```

where:

localFile
is the name of the file to be copied from the local host.

remoteFile
is the name of the copy on the remote host.

"**Binary**"
sets the file transfer mode to binary or image.

"**Ascii**"
sets the file transfer type to ASCII (flat text).

The return value is an FTP error code.

FtpLs()

The FtpLs() call gets the directory information about the current directory of the remote FTP server. The directory information is in short format and placed in the stem variables.

The remote host running the FTP server is specified with the FtpSetUser() call.

The FtpLs() call is similar to the FtpDir() call, which gets the directory information in the long format.

Syntax:

```
rc = FtpLs(pattern,stem)
```

where:

pattern
is the name or pattern of the files to be listed on the remote host. Patterns are any combination of ASCII characters. The following characters have special meanings:

***** stands for any character or group of characters.

? stands for any single character.

stem
specifies the variable *stem*. The stem string must end with a period (.).

When the function completes, the variable *stem.0* is set to the number of items returned in the stem variable. Each item of the stem variable contains directory information, except the first, which contains the number of items.

Example:

```
rc=FtpLs("ftpx*.c","files.")
```

locates three files in the current directory of the remote FTP server. The stemmed variables are:

```
Variable       Value
files.0        3
files.1        ftpxdir.c
files.2        ftpxren.c
files.3        ftpxdel.c
```

The return value is an FTP error code.

FtpDir()

The FtpDir() call gets the directory information about the current directory of the remote FTP server. The directory information is in long format and placed in the stem variables.

The form of the output depends on the operating system running on the FTP server machine. The output look similar to that given in the Example in the description of the FtpLs() function, but with additional information.

The remote host running the FTP server is specified with the FtpSetUser() call.

The FtpDir() call is similar to the FtpLs() call, which gets the directory information in the short format.

Syntax:

```
rc = FtpDir(pattern,stem)
```

where:

pattern
is the name or pattern of the files to be listed on the remote host. Patterns are any combination of ASCII characters. The following characters have special meanings:

***** stands for any character or group of characters.

? stands for any single character.

stem
specifies the variable *stem*. The stem string must end with a period (.).

When the function completes, the variable *stem.0* is set to the number of items returned in the stem variable. Each item of the stem variable contains directory information, except the first, which contains the number of items.

The return value is an FTP error code.

FtpChDir()

The FtpChDir() call changes the working directory for the remote FTP server. The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpChDir(directory)
```

where:

directory
is the new working directory name for the remote FTP server.

The return value is an FTP error code.

FtpMkDir()

The FtpMkDir() call creates a new directory on the remote host. The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpMkDir(directory)
```

where:

directory
specifies the name of the directory to be created on the remote host.

The return value is an FTP error code.

FtpRmDir()

The FtpRmDir() call removes a directory from the remote host. The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpRmDir(directory)
```

where:

directory
specifies the name of the directory to be removed from the remote host.

The return value is an FTP error code.

FtpPwd()

The FtpPwd() call (print working directory) gets the name of the current working directory of the remote FTP server and places it in the variable *dirName*. The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpPwd(dirName)
```

where:

dirName
returns the name of the current working directory of the remote FTP server.

The return value is an FTP error code.

FtpQuote()

The FtpQuote() call sends a string to the remote FTP server. The remote FTP server must support the FTP command contained in this string.

The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpQuote(quote<, replyName>)
```

where:

quote
is the string to be sent to the remote FTP server.

replyName
is the variable to which the full reply of the FTP server to the quoted command is copied.

The return value is an FTP error code.

FtpSite()

The FtpSite() call sends a string to the remote FTP server, in a situation where a service that is not covered by the File Transfer Protocol is needed. Your server system must support this information.

The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpSite(site)
```

where:

site
is the string to be sent to the remote FTP server.

The return value is an FTP error code.

FtpSys()

The FtpSys() call returns a description of the operating system running on the remote host.

The remote host running the FTP server is specified with the FtpSetUser() call.

Syntax:

```
rc = FtpSys(operSys)
```

where:

operSys
is the FTP server description of the operating system running on the remote host.

If the call is successful, the description of the remote operating system is returned. If the call fails, an FTP error code is returned.

FtpProxy()

The FtpProxy() call copies a file from one remote FTP server to another. You can use different file names on each host. Optionally, you can specify the transfer to occur in binary or ASCII mode.

Syntax:

```
rc = FtpProxy(host1,userid1,password1,account1,
              host2,userid2,password2,account2,
              file1,file2<,"Binary"|"Ascii">)
```

where:

host1
identifies the remote FTP server where the copy is to reside.

userid1
identifies the user to the remote FTP server where the copy is to reside.

password1
supplies the password to the remote FTP server where the copy is to reside.

account1
supplies host-dependent accounting information to the remote FTP server where the copy is to reside. Use “NULL” if there is no account information.

host2
identifies the remote FTP server containing the file to be copied.

userid2
identifies the user to the remote FTP server containing the file to be copied.

password2
supplies the password to the remote FTP server containing the file to be copied.

account2
supplies host-dependent accounting information to the remote FTP server containing the file to be copied. Use "NULL" if there is no accounting information.

file1
identifies the name of the copy.

file2
identifies the name of the file to be copied.

"**Binary**"
sets the file transfer mode to binary or image.

"**Ascii**"
sets the file transfer type to ASCII (flat text).

The return value is an FTP error code.

FtpPing()

The FtpPing() call sends a ping to the remote host to resolve the host name through the use of a name server. If there is no name server, the FtpPing() call searches the hosts file in the ETC directory for a matching host name.

The user must have administrator rights to execute this function.

Syntax:

```
rc = FtpPing(host,length)
```

where:

host
identifies the name of the remote host.

length
identifies the length of the ping packets.

Return values:

If no error occurs, the FtpPing() call returns the number of milliseconds required for the echo to return. If an error occurs, one of the following values is returned:

–1 A general FTP function call error occurred. Refer to the FTP error codes on how to obtain further error information.

PINGHOST
Unknown host specified.

PINGPROTO
ICMP protocol not handled by TCP/IP stack.

PINGRECV
No echo received.

PINGREPLY
The host does not reply.

PINGSEND
No data sent.

PINGSOCKET
Unable to create socket.

Note: Depending on the resolution of the system timer used to determine the return time a result of 0 milliseconds might be returned. This indicates that the echo was returned in less than the smallest measurable time.

FtpTrace()

Switches on or off a trace operation on the REXX FTP procedure. This function acts as a simple toggle. If the trace operation is switched on, the FTP API functions and their parameters are displayed.

Syntax:

```
rc = FtpTrace()
```

Return value:

0 The trace function is off.

1 The trace function is on.

FtpTraceLog()

Opens a log file and writes to it the messages that were traced between the FTP client and server. If the function is called again, the open log file is closed and another is opened; this can be either a different file or the same one. This procedure can be used to record only the latest transfer, keeping the log file small and recording only the latest errors.

Syntax:

```
rc = FtpTraceLog(logFileName<,logFileMode>)
```

Parameters:

logFileName
The name of the log file, or the fully-qualified path and the name of the log file.

logFileMode
The mode of the log file.

1 Overwrites the previous log file.

2 Appends information to the previous log file. This is the default.

Return value:

0 The trace function has been successfully started.

–1 The trace function has failed to start.

FtpTraceLogOff()

Ends the writing of the log file and closes it.

Syntax:

```
rc = FtpTraceLogOff()
```

Return value:

0 The trace function has been successfully stopped.

1 The trace function was already stopped.

Appendix A. A Sample RxFtp Program

The following is a sample program using REXX FTP library functions.

```
/*============================================================*/
/*   Basic RxFTP sample to send a file with cmd/reply logging   */
/*============================================================*/

/* Define the variables for:                                  */
     server = '127.0.0.1'           /* IP address or server name   */
     userid = 'remote_user_ID'
     passwd = 'password_of_remote_user'
     trclog = 'logfile.txt'         /* Trace log file name         */
     rc     = 0                     /* Set return code to 0        */

/* Ensure that the function package is properly registered:        */
If RxFuncQuery('FtpDropFuncs') then
  Do
    rc = RxFuncAdd('FtpLoadFuncs', 'rxftp', 'FtpLoadFuncs')
    rc = FtpLoadFuncs()
  End

If rc <> 0 then
 Do
   Say ' *** FTP functions could not be loaded.'
   EXIT rc
End

/* Start tracing FTP commands and logging of replies              */
rc = FtpTrace()
rc = FtpTraceLog( trclog, '1')
If rc = 0 then
  Say ' Replies will be written to log file: 'trclog'.'
Else
  Say ' No writing to log file: 'trclog' possible.'

/* Define remote host and user to be used during the session       */
rc = FtpSetUser(server, userid, passwd)
If rc = 1 then
  Say ' Settings for a connection are stored.'
Else
  Call Terminate " *** Settings for connection failed."

/* S/390 FTP servers need to be put in active mode                 */
rc = FtpSetActiveMode('1')
If rc = 1 then
  Say ' Active mode has been set.'
Else
  Call Terminate " *** Active mode setting has failed."

/* Transfer an ASCII file to the remote ftp server                */
```

```
rc = FtpPut('sample.rex', 'sample.put', 'ASCII')
If rc = 0 then
  Call Terminate ' File has been sent. '
Else
  Call Terminate " *** File has NOT been sent."

/* Terminate the file transfer                                    */
Terminate:
  Parse Arg Message

  Say Message

  Call FtpLogoff   /* Log off (sends QUIT command to the server)    */

  Call FtpTraceLogOff  /* Stop logging - close log file.           */
  Call FtpTrace

  Call FtpDropFuncs    /* Remove the function package.             */

EXIT rc
```

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will

be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland
Informationssysteme GmbH
Department 3982
Pascalstrasse 100
70569 Stuttgart
Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement or any equivalent agreement between us.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following term is a trademark of the IBM Corporation in the United States, other countries, or both:

IBM

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.