



Object REXX for Linux

# REXX FTP Library Functions (RxFTP)

*Version 1.2*

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Appendix. Notices" on page 21.

**First Edition, March 1999**

This edition applies to Version 1.2 of IBM® Object REXX for Linux, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

© Copyright International Business Machines Corporation 1999. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

## Contents

<b>Chapter 1. What Is RxFtp?</b> . . . . .	<b>1</b>	FtpRename() . . . . .	11
<b>Chapter 2. Installation and Removal.</b> . . . .	<b>3</b>	FtpGet() . . . . .	12
<b>Chapter 3. Return Values</b> . . . . .	<b>5</b>	FtpPut() . . . . .	12
Set Error Codes . . . . .	5	FtpPutUnique() . . . . .	13
FTP Error Codes . . . . .	5	FtpLs() . . . . .	14
<b>Chapter 4. Functions</b> . . . . .	<b>7</b>	FtpDir() . . . . .	15
FtpLoadFuncs() . . . . .	7	FtpChDir() . . . . .	16
FtpDropFuncs() . . . . .	8	FtpMkDir() . . . . .	16
FtpVersion() . . . . .	8	FtpRmdir() . . . . .	16
FtpSetUser() . . . . .	8	FtpPwd() . . . . .	17
FtpSetActiveMode() . . . . .	9	FtpQuote() . . . . .	17
FtpSetBinary() . . . . .	9	FtpSite() . . . . .	17
FtpLogoff() . . . . .	10	FtpSys() . . . . .	18
FtpAppend() . . . . .	10	FtpProxy() . . . . .	18
FtpDelete() . . . . .	11	FtpPing() . . . . .	19
		<b>Appendix. Notices</b> . . . . .	<b>21</b>
		Trademarks and Service Marks . . . . .	23



---

## 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. The RxFtp function names are similar to the 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 the TCP/IP support to be active on your system.



---

## Chapter 2. Installation and Removal

The Rxftp package is contained in the file **librxftp.so**. This library must be placed in a directory listed in your LIBPATH. To get access to the functions in the Rxftp package, execute the following REXX code:

```
rc = RxFuncAdd("FTPLoadFuncs","rxftp","FTPLoadFuncs")  
rc = FtpLoadFuncs()
```

To unload the library, call the FtpDropFuncs() function.





---

## Chapter 3. Return Values

Return values can either be set errors or FTP errors.

---

### Set Error Codes

The `FtpSetUser()`, `FtpSetActiveMode()`, and `FtpSetBinary()` functions return a set error code. It can have one of the following values:

- 1**        if the string passed is valid
- 0**        if the string passed is not valid

---

### FTP Error Codes

All `RxFtp` functions, except `FtpSetUser()`, `FtpSetActiveMode()`, and `FtpSetBinary()`, 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

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()

---

### 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.

The return value is an FTP error code.

---

## **FtpDropFuncs()**

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

Syntax:

```
rc = FtpDropFuncs()
```

The return value is an FTP error code.

---

## **FtpVersion()**

The FtpVersion() call identifies the version of the RxFTP package, which is currently 2.0.

Syntax:

```
rc = FtpVersion(version)
```

where:

*version*

is the version of the RxFTP program that you are running.

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

---

## **FtpSetUser()**

The FtpSetUser() call sets the host name, user ID, password, and, optionally, the user's account for the remote host. These parameters are kept during the entire FTP session.

To prevent unauthorized access to the remote host, 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 host.

*account*

supplies host-dependent accounting information.

The return value is a set error code.

---

## **FtpSetActiveMode()**

The `FtpSetActiveMode()` call sets the default transfer mode for the FTP protocol. Default is the passive transfer mode because it is supported 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. In this case, you can explicitly select the active transfer mode for the current session. The active mode is used until you reset this mode with another `FtpSetActiveMode()` call. Note that in active mode you cannot transfer files through a firewall.

Syntax:

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

where:

**"1"**

sets the file transfer mode to active mode only.

**"0"**

sets the file transfer mode to passive with fallback to active mode.

The return value is a set error code.

---

## **FtpSetBinary()**

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

The `FtpSetBinary()` call can be passed as an abbreviation. For "Binary", you can also specify "b" or "BIN". For "Ascii", you can specify "as".

Syntax:

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

where:

**"Binary"**

sets the file transfer mode to binary or image.

**"Ascii"**

sets the file transfer mode to ASCII (flat text).

The return value is a set error code.

---

## FtpLogoff()

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

The remote host 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 host 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 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 a remote host. The remote host 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 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 host to the local workstation. 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 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 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.

---

## FtpPut()

The FtpPut() call copies a single file from the local workstation to a remote host. 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 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 workstation.

*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 workstation 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 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 workstation.

*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 host. The directory information is in short format and placed in the stemmed variables.

The remote host 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 stem variables returned. The directory information is set in all following variable stems. Because the period is not appended automatically, you must include it in the stem.

Example:

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

located three files in the current directory of the remote host. 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 host. The directory information is in long format and placed in the stemmed variables.

The remote host 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 stem variables returned. The directory information is set in all following variable stems. Because the period is not appended automatically, you must include it in the stem.

Example:

```
rc=FtpDir("ftp*.c","files.")
```

located three files in the current directory of the remote host. The stemmed variables are:

Variable	Value
files.0	3
files.1	-r-xr-xr-x 1 bin bin 1215187 Apr 11 1996 ftpxdir.c
files.2	-r-xr-xr-x 1 bin bin 39275 Mar 08 1996 ftpxren.c
files.3	-r-xr-xr-x 1 bin bin 312461 Apr 11 1996 ftpxdel.c

The return value is an FTP error code.

---

## FtpChDir()

The FtpChDir() call changes the working directory on the remote host. The remote host is specified with the FtpSetUser() call.

Syntax:

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

where:

*directory*

is the new directory name on the remote host.

The return value is an FTP error code.

---

## FtpMkDir()

The FtpMkDir() call creates a new directory on the remote host. The remote host 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 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 host and places it in the variable *dirName*. The remote host is specified with the FtpSetUser() call.

Syntax:

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

where:

*dirName*

returns the name of the current working directory of the remote host.

The return value is an FTP error code.

---

## FtpQuote()

The FtpQuote() call sends a string to the remote host. Your server must support the information contained in this string.

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

Syntax:

```
rc = FtpQuote(quote)
```

where:

*quote*

is the string to be sent to the remote server.

The return value is an FTP error code.

---

## FtpSite()

The FtpSite() call sends FTP information to the remote host. Your server must support this information.

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

Syntax:

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

where:

*site*

is the string to be sent to the remote host.

The return value is an FTP error code.

---

## FtpSys()

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

The remote host 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.

The return value is an FTP error code.

---

## FtpProxy()

The FtpProxy() call copies a file from one remote host 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 host where the copy is to reside.

*userid1*

identifies the user to the remote host where the copy is to reside.

*password1*

supplies the password to the remote host where the copy is to reside.

*account1*  
 supplies host-dependent accounting information to the remote host where the copy is to reside. Use "NULL" if there is no account information.

*host2*  
 identifies the remote host containing the file to be copied.

*userid2*  
 identifies the user to the remote host containing the file to be copied.

*password2*  
 supplies the password to the remote host containing the file to be copied.

*account2*  
 supplies host-dependent accounting information to the remote host 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).

Return values:

**0**        if the call was successful  
**-1**       if an error occurred during the call

---

## **FtpPing()**

The FtpPing() call sends a ping to the remote host to resolve the host name through 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.

Only the root user can perform this call.

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.



---

## Appendix. 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 and Service Marks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM

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

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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