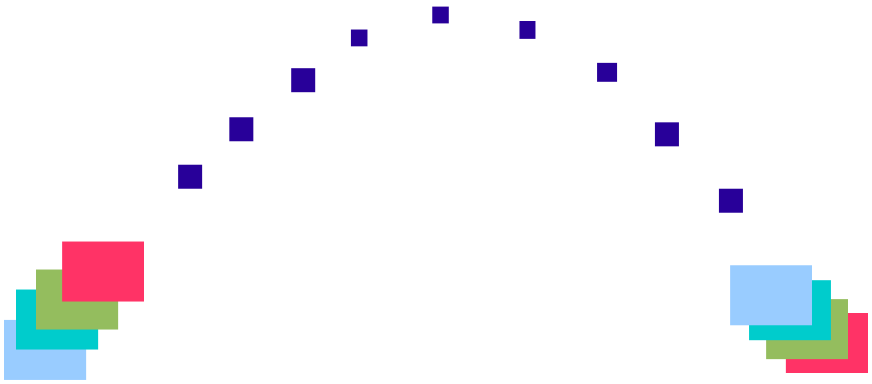
Z M C S

ZMODEM File Server and Client

DOS, Linux, Windows



Version 5

LICENSE AGREEMENT

ZMODEM FILE SERVER and CLIENT - ZMCS (software)

CAREFULLY READ THE LEGAL AGREEMENT, CORRESPONDING TO THE LICENSE YOU PURCHASED, WHICH SETS FORTH THE GENERAL TERMS AND CONDITIONS FOR THE USE OF THE LICENSED SOFTWARE. IF YOU DO NOT AGREE TO ALL OF THE TERMS AND CONDITIONS SET FORTH IN THE AGREEMENT, DO NOT INSTALL THE SOFTWARE. IF APPLICABLE, YOU MAY RETURN THE PRODUCT TO THE PLACE OF PURCHASE FOR A REFUND. THE PARTIES CONFIRM THAT IT IS THEIR WISH THAT THIS AGREEMENT HAS BEEN WRITTEN IN THE ENGLISH LANGUAGE ONLY.

LICENSE TERMS
Anybody reproducing this "software" program will be prosecuted. The rights for the "software" documentation and software program remain with ADONTEC Ltd.

Only one single copy can be made for back-up purposes only. The software program may not be used simultaneously by different persons at different locations and on different machines (PC), just as it is impossible that one book is read at the same time in different locations by several persons.

Each license of this "software" enables the use of this "software" on one PC.

The software may store license information on Hard-disk. Removing or manipulating such information or manipulating, reverse engineering the software itself is prohibited and voids license.

Illegal use of this software will be prosecuted.

VOLUME LICENSE
A volume license product includes the same amount of licenses as unique serial-numbers.

A volume license product can be installed on a number of PC according to the number of serial-numbers supplied. Each single copy can be installed on one PC using a unique serial-number. The same serial-number should not be used twice. You must have a reasonable mechanism in place to ensure that the number of PC on which the software has been installed does not exceed the number of licenses you have optained.

Illegal use of this software will be prosecuted.

GUARANTEE LIMITATION
We always make every effort to supply to you a perfect software. We would like to bring to your attention though that with the present level of technology we cannot quarantee that the software program works interrupt and error-free in all combinations and applications.
No quarantee is given for the correctness of the contents of this manual since errors, inspite of all efforts, never can be completely avoided. For this reason we are grateful anytime for comments.
The liability for indirect damages, direct damages, follow on damages and damages to third parties is excluded within the legal framework. The liability through gross negligence and intent is hereby unaffected, in each case though the liability is limited to the purchase price.

TRADE MARKS
Microsoft Windows are registered trade marks of the Microsoft Corporation.
Other brand and product names are trademarks or registered trademarks of their respective holders.

Support and Registration

Thank you for purchasing and uisng this Software By registering you are eligible for free technical support and future savings on upgrades.

Please use the forms at http://www.adontec.com for registration, support and other conctacts.

Content

ZMODEM File Server and Client (ZMCS)

What is the purpose of this software ?
ZMCS *(ZMODEM Client/Server)* is a console program which transfers files using the ZMODEM protocol or the KERMIT protocol

ZMCS can establish connections* using a *Modem* or a *Direct Link* (serial cable) or ISDN or TCP/IP and transfer files.

ZMCS can act as server or client. In *server mode* it waits for a client to connect and handles the requests for sending or receiving files. In *client mode* it connects to a server and sends or retrieves files to or from the server.

In *client mode* it runs unattended until all files are sent or received. It then ends autonomous. In s*erver mode* it accepts client connections, receives files, transmits files and may be terminated, if no more needed, by an application or by the user.

ZMCS is no island solution. Both, the client and the server can communicate with third party software (e.g. Hyperterminal). So access is not restricted between ZMCS application but third party software can easily connect to the ZMCS Server and the ZMCS Client can connect to third party server when no other special specifications requiered (like login protocols etc.).

ZMCS also enables to add easily file transmission capabilities to any existing application.

How is it best started ?
ZMCS is usually started with a Batch file, a Script, Windows Autostart or from within an Windows application and runs completely and unattended in background. The client ends when done. The server runs endless until closed by pressing the key 'E' or ESC or by closing it from within a user application.

The software provides status information. All status information and printouts can be routed into a log file (informational mode) or disabled (quiet mode). A low level data trace function is offered as well.

ZMCS can also run completely hidden and in background without any visible window.

***different executbales (license) support Serial or TCP/IP or ISDN.**

What commands and options does it offer ?
The ZMCS software is driven by command line parameters:

Syntax

SERIAL

```
  ZMCS.EXE /M<r|t|s|ux>[H] [/Cno] [/Tphone] [/Ffilename|/F@filelistname]
      [I[r]] [/X[m|c]] [/XP[Z[R|r|..]|K]] [/XL[A][U=username,P=password] [XI[p|c]]
```

TCP/IP

```
  ZMCS.EXE /M<r|t|s|ux>[H] [/Cno] [[/Tip:port]|[/Eip:port]]
[/Ffilename|/F@filelistname] [I[r]] [/X[m|c]] [/XP[Z[R|r|..]|K]] [/XL[A]
[U=username,P=password] [XI[p|c]]
```

ISDN

```
  ZMCS.EXE /M<r|t|s|ux>[H] [/Cno] [[/Tphone]|[/Emsn]] [/Ffilename|/F@filelistname]
      [I[r]] [/X[m|c]] [/XP[Z[R|r|..]|K]] [/XL[A][U=username,P=password] [XI[p|c]]

   /M  defines the run mode r=receiver, t=transmitter, s=server,
       u=unload the server running on COMx,
       H=disable Handshake* (allows third party programs to connect)

   /C  defines the COMM channel number, no=[1,2,...,255]
   /C  defines a communication number, no=[1,2,...,255]
   /C  defines a communication number, no=[1,2,...,255]

   /T  defines the phone number to dial
   /T  defines the IP and Port address to connect to (client)
   /E  defines the IP and Port address to listen to (server)
   /E  defines the MSN to listen to (server)

   /F  defines the file name to send or receive
        file specifications e.g. data.txt, acc*.txt, *.doc, *.A??
   /F@ defines a Job File
   /I[r] ignore signals and conditions
   /X[c|m]  defines the file copy mode  c=copy, m=move
        in case of 'm' files are deleted after successfully transfered
   /XP[Z[R|r|..]|K] defines the protocol and options to use (Z=ZMODEM, K=KERMIT)
   /XL[A][U=username,P=password] activates the login procedure
   /XI[p|c] defines extended informational flags
   /L  direct link, no modem dialup involved
   /Q  quiet mode, suppress messages
   /W[working dir]  defines the working directory

  additional parameters
   /Bbaudrate /Ddatabits /Sstopbits /Pparity
      baudrate=[50-115200] databits=[5-8] stopbits=[1|2] parity=[N|O|E|M|S]

[] = optionally, | = use one of the available values
```

* This option is only supported to be able to run with third party ZMODEM software. Please note that in this case the ZMCS handshake is not performed and the functionality is limited to standard file transfer.

Examples for Serial

Sample 1:

```
 1. Start server on COM1 (ttyS0):  ZMCS /C1 /Ms

 2. Client, transmit (copy) files: ZMCS /C2 /Mt /T123456789 /Fsales.dat
 3. Client, retrieve (copy) files: ZMCS /C2 /Mr /T123456789 /Fleads.dat
 4. Client, retrieve (copy) files: ZMCS /C2 /Mr /T123456789 /Forders.dat

 or 2. to 4. within one session using a Job File
 2. Client, job file (copy): ZMCS /C2 /Mt /T123456789 /F@job001.txt
```

Sample 2:

```
 1. Start server on COM3 (ttyS2): ZMCS /C3 /Ms /L
    This servers is running on COM3 (ttyS2) and accepts direct cable connections.
    There is no modem involved.

 2. Client, transmit (copy) files: ZMCS /C2 /Mt /L /Fmeasure1239.mdb

 3. Unloading the server on COM3 (ttyS2): ZMCS /Mu3
```

Sample 3:

```
 1. Start server on COM3 (ttyS2): ZMCS /C3 /Ms /XL
    This server expects connections on COM3
    Clients must authenticate himselfs (login) using username and password.

 2. Client, transmit (move): ZMCS /C2 /Mt /T123456789 /Xm
                                    /XLU=dorian,P=secret /F*.pdf
    Clients authenticates as dorian,secret.

 3. Unloading the server on COM3 (ttyS2): ZMCS /Mu3
```

Sample 4:

```
 1. Start server on COM1 (ttyS0): ZMCS /C1 /MsH /XPZR
    This server allows third party communication software to connect without
    issuing the ZMCS connecting dialog.
    ZMODEM "crash recovery" enabled.

 2. As client one can use a third party program (e.g. Hyperterminal) to transmit
    files to the server using ZMODEM protocol.
```

* First serial /C1 is COM1 under DOS or Windows or ttyS0 under Linux.

Examples for ISDN

Sample 1:

```
 1. Start server on communication number 1:  ZMCS /C1 /Ms /E90020
    This server expects connections on MSN 90020

 2. Client, transmit (copy) files: ZMCS /C2 /Mt /T90020 /Fsales.dat
 3. Client, retrieve (copy) files: ZMCS /C2 /Mr /T90020 /Fleads.dat
```

Sample 2:

```
 1. Unloading the server on communication number 1: ZMCS /Mu1
```

Sample 3:

```
 1. Start server on communication number 1: ZMCS /C1 /Ms /E90020 /XL
    This server expects connections on phone number 90020 (MSN)
    Clients must authenticate himselfs (login) using username and password.

 2. Client, transmit (move): ZMCS /C2 /Mt /T90020 /Xm
                                  /XLU=dorian,P=secret /F*.pdf
    Clients authenticates as dorian,secret.

 3. Unloading the server on communication link 1: ZMCS /Mu1
```

Sample 4:

```
1. Start server on communication number 1: ZMCS /C1 /MsH /XPZR
   This server expects connections on any available phone number (MSN).
   This server allows third party communication software to connect without
   issuing the ZMCS connecting dialog.
   ZMODEM "crash recovery" enabled.

2. As client one can use a third party program to transmit files to the server
   using ZMODEM protocol.
```

Examples for TCP/IP

Sample 1:

1. Start server on communication number 1: ZMCS /C1 /Ms /Ewww.mysite.com:9100
   This server expects connections on „www.mysite.com:9100".

2. Client, transmit (copy) files: ZMCS /C2 /Mt /Twww.mysite.com:9100 /Fsales.dat
3. Client, retrieve (copy) files: ZMCS /C2 /Mr /Twww.mysite.com:9100 /Fleads.dat

Sample 2:

1. Unloading the server on communication number 1: ZMCS /Mu1

Sample 3:

1. Start server on communication number 1: ZMCS /C1 /Ms /Ehost:9100 /XL
   This server expects connections on „host:9100".
   Clients must authenticate himselfs (login) using *username* and *password*.

2. Client, transmit (move): ZMCS /C2 /Mt /Thost:9100 /Xm
   /XLU=dorian,P=secret /F*.pdf
   Clients authenticates as *dorian,secret*.

3. Unloading the server on communication link 1: ZMCS /Mu1

Sample 4:

1. Start server on communication number 1: ZMCS /C1 /MsH /XPZR
   This server expects connections on "host:9000".
   This server allows third party communication software to connect without issuing the ZMCS connecting dialog.
   ZMODEM "crash recovery" enabled.

2. As client one can use a third party program to transmit files to the server using ZMODEM protocol.

Command Line Parameters

Most parameters are self explained. In the following the most important parameters will be presented with details.

Parameter /C

Defines the serial port to use (1=first serial port e.g. COM1 or ttyS0, 2=second serial port e.g. COM2 or ttyS1, etc.) e.g. `/C2`.
With ISDN and TCP/IP this parameter defines a communication number (1 to 255) used to refer at e.g. when need to unload a server.

Parameter /E

The parameter **/E** specifies additional information for the server:

Serial: No function.

ISDN: The msn a server should listen at. If none set all available MSN* will be accepted.
e.g. /e90020

TCP/IP: The ip and port address a server should accept client connections.
e.g. /e192.168.2:9000
When setting IP address to „host“ it will use the PC IP address.
e.g. /ehost:9000
If nothing set it will listen to „host:9000“.

Note: ISDN and TCP/IP currently Windows only.
*MSN = Multiple Subscriber Number.

Parameter /F

The parameter **/F** specifies a file name and may contain a relative or a complete path
e.g. `/Fc:\data\db01.txt` (complete path).
`/Fdata\db01.txt` (relative path).
A relative path is always within the working directory. A file name can also contain wildcards (*,?). If the file name contains spaces it must be quoted with ""
e.g. `/F"d:\temp\My Files\*.txt"`.

Quoted parameters may fail in the DOS version or ZMCS.EXE.

Parameter /F@ - Job File

A *Job File* is similar to a script file. It contains commands to be executed while a client is connected. A job file defines a list of files, that can be transmitted or received within one session. This list of files is stored within a text file. Using a job file it's possible to transmit and retrieve files within one session thus avoiding a re-connect.

A job file is specified with the parameter **/F@**. The character '@' is hereby used to distinguish a normal file from a job.

e.g. `/F@"d:\temp\My Files\jobfile.txt"`

If no path included the file is searched in the same directory where ZMCS.EXE resides.

A *Job File* contains commands and file names. The Job file also supports "sections". Currently a "TX" and a "RX" section for files are supported. If no section specified a "TX" section is assumed and all files are transmitted. "RX" marks files to be retrieved.

Sample job file: job001.txt

```
# Sales man 001
#RX
leads.dat
orders.dat
#TX
```

sales.dat

A *Job File* is a text file and can be created using a simple editor. Every entry must start in the first column. A command begins with '#'. A file name is entered as is. Comments must start with '#' followed by a space.
Only clients can handle job files. In order to handle a job file the client can be started with /Mr or /Mt. The optional parameter **H** (e.g. **/MsH**) should not be used else the job file might not execute as expected if the *ZMCS synchronization protocol* is disabled.
When using the **/MH** switch the job file can only execute requests that are automatically known by the remote since the ZMCS can no more instruct the remote (e.g. can't request a specific file but it can receive files automatically sent by the remote).

Parameter /I

Ignore signal and conditions. Currently option 'r' can be used to ignore false ringing signals reported by some modem or driver. e.g. **/Ir**

Parameter /L

Indicates a local connection by serial cable. No modem involved.
Ignored with ISDN or TCP/IP.

Parameter /M

Defines the execution mode.

- `/Mr` defines a Client that will retrieve files from a Server.
- `/Mt` defines a Client that will transmit files to a Server..
- `/Ms` defines a Server.
- `/Mux` unloads the Server related to the communication link `x` as defined with parameter `/C`

The additional Parameter **H** (e.g. **/MsH**) disables the *ZMCS synchronization protocol* (Handshake) allowing third party application to connect and transmit files to the server.

Parameter /T

Serial & ISDN:

Defines the phone number to connect to e.g. `/T5556667890`.

TCP/IP:

Defines the IP & Port address to connect to
e.g. /Twww.mysite.com:9000
When setting IP address to „host“ it will use the PC own IP address
e.g. /Thost:9000
If port omitted 9000 will be used.

Note: ISDN and TCP/IP currently Windows only.

Parameter /W

ZMCS can be configured to accept files only in the current working directory (e.g. **`/Wd:\data`**). Files to be transmitted or retrieved must be located inside the working directory. Files outside the working directory are ignored. Thus protecting other directories.
Any relative path used in the parameter **/F** are resolved inside this working directory.
If no working directory specified the local directory where ZMCS.exe was started is taken. *Any specified working directory must exist and not be marked as readonly.* ZMCS does not create missing directories.
If the path contains spaces it must be quoted with "" e.g. `/W"d:\My Data\My Files"`.
Quoted parameters may fail in the DOS version or ZMCS.EXE.

Parameter /X

The parameter **/X** used in conjunction with option **C** or **M** controls the file copy mode.
With **/Xc** files are transmitted in *copy mode*.
With **/Xm** files are transmitted in *move mode,* thus the original files **are deleted** after successfully transmitted.

Parameter /XI

Provides additional flags used to control the information as reported. Currently the option **P** defines the progress printout to be in percent e.g. [.....], where each dot (1 to 10) represents approximately 10%. Activate using `/Xip`.
The default (when this option omitted) shows transfered data blocks.
The next option offered is **C** which activates Interprocess Communication (IPC). IPC is supported through Mailslots. The used Mailslot is by ZMCS is „\\.\mailslot\zmcs_c_xx“ where xx=01 with COM1, 02 with COM2 etc.). A small IPC sample written in C/C++ is available. Activate using `/XIc`.
In order to activate more than one option just add the required options after the „/XI“ e.g. /Xipc.

Parameter /XL

The server can be configured to accept only authenticated user. The parameter **/XL** can be used to activate the login procedure on the server. Once set it expects from every client to authenticate itself by entering *username* and *password*. This mechanism also allows to specify a per user working directory. Additionally but optionally, a computers user confirmation can be requested on every client login (**/XLA**).

```
Server: /XL
```

Instructs the server to activate the login procedure and to load user information from the file `ZMCS_usr.ini`. The file `ZMCS_usr.ini` is expected in the servers directory.

Sample *ZMCS_usr.ini*:

```
[dorian]
PASS=secret01
WORKDIR=dorian_data
```

For every user a section should be created with keys 'PASS' and 'WORKDIR'. The key 'WORKDIR' is optional and defines the per user working directory. If 'WORKDIR' is not defined the working directory from the parameter /W will be used instead.

The above sample defines the user 'dorian' with password 'secret01' and a relative working directory 'dorian_data'. If the working directory is specified without absolute path, it defines a subdirectory starting relatively to the directory of ZMCS.EXE.

The parameter **/XLU=*username*,P=*password*** can be used to activate the login procedure on the client side.

```
Client: /XLU=dorian,P=secret01
```

Instructs the client to start the login procedure with username 'dorian' and password 'secret01'.

Manual Login Control

Every client login is checked against a *Username* and *Password*. Additionaly the computer user can request from the server to manually check every client login. To enable this *manual user access control* the server needs to be started with the parameter /**XLA**. After that it shows a confirmation dialog to the computer user after a client attempts to login. The computer user should now confirm or reject within 8 seconds in order to allow the login procedure to continue normally.

Parameter /XP

The parameter **/XPK** activates the *KERMIT protocol*. The parameter **/XPZ** activates the *ZMODEM protocol*, which is also the default one. When ZMCS is running on both sides, switching protocol on one ZMCS will force the same protocol on the other ZMCS. In order this to work one should not disable the *ZMCS synchronization protocol* (/MH).

Communicating with a third party software usually requires to disable the *ZMCS synchronization protocol* since it is unlikely to be known (/MH). And, since there is no common way for ZMCS to tell a third party software to use a specific protocol, one must preconfigure the used software.

The parameter **/XPZ** also supports optional ZMODEM *file options* /XPZ[<R|r><S><N|O|P|X><->]. File options are mainly directions from the transmitter to the receiver on how to handle the received file. When the sender does not transmit file options the receiver will use it's own, if any defined.

The following file options currently supported:

R	Enable the ZMODEM crash recovery feature.
r	Disable the ZMODEM crash recovery feature.
S	Skip if absent: Accept the file only if it exists, else nothing transferred.
N	Accept the file only if it is newer or longer.
O	Overwrite: The file is always transferred (=refresh).
P	Protect destination file, if it exists, else transfer/receive as new.
X	Extend: Appends the data to the destination file, if it exists, else transfer/receive as new.
-	Ignore any file options received and use the local one.

Only one option of each block is allowed!
If nothing specified the ZMODEM protocol **defaults** to **/XPZr.** Thus it will not resume an interrupted file transfer but transmit the file as new and it will transfer files that are new or newer (only date checked, length is not checked).

The ZMODEM *file options* work independently of the option /MH.

Examples

```
ZMODEM without „crash recovery".
    On receiver side /XPZr-  or  on sender side  /XPZr

ZMODEM with „crash recovery".
    On receiver side /XPZR-  or  on sender side  /XPZR

ZMODEM without „crash recovery" and protect destination file.
    On receiver side /XPZP-  or  on sender side  /XPZP

ZMODEM without „crash recovery" and skip if absent.
    On receiver side /XPZS-  or  on sender side  /XPZS

ZMODEM without „crash recovery" and append new data.
    On receiver side  /XPZX-  or  on sender side /XPZX
```

In order to `extend` the file, `on receiver side,` with the new data only, `the new file should contain only the new data and a new date.`

```
ZMODEM without „crash recovery" and files always transferred.
    On receiver side  /XPZO-  or  on sender side  /XPZO
    or on server side  /XPZO-  (runs for both sides because of '-' !)

ZMODEM with „crash recovery" and files transferred only when exists on receiver
side.
    On receiver side  /XPZRSN-  or  on sender side  /XPZRSN
    or on server side  /XPZRSN-  (runs for both sides because of '-' !)
```

Notes:

The *KERMIT protocol* and ZMODEM *file options* are supported in ZMCS for Windows only.

Since some client fail to provide the correct file size, the option /XPZR should be used with checked clients only.

The program defaults are

The ZMCS uses the following defaults, if not specified else:

Serial
`/C2 /B38400 /D8 /S1 /Pn /Xc /XPZr` (COM2, 38400,8,1,N, copy files, ZMODEM)

ISDN
`/C2 /Pn /Xc /XPZr` (all MSN, copy files, ZMODEM)

TCP/IP
`/C2 /Pn /Xc /XPZr` („host:9000“, copy files, ZMODEM)

Canceling the server

The program can be terminated by pressing the 'E' or ESC key.
From within a third party application by using the parameter **/Mux** (x=1, 2, ... COM1, COM2, ...) :

For example, to unload the server on COM3: `ZMCS /Mu3`

Job File

A *Job File* is similar to a script file and it contains commands to be executed while a client is connected. A job file is expected to be a pure text file in ANSI (8-Bit) format.

A job file may contain commands to transfer data and files, branches, control logical sequences and script flow, setup connection, perform anytime manual login, protocol switch, run loops, set variables, and many more.

A job file is specified with the parameter **/F@**. The character '@' is hereby used to distinguish a normal file from a job file.

e.g. `/F@"d:\temp\My Files\jobfile.txt"`

If no path included the file is searched in the same directory where ZMCS.EXE resides.

A *Job File* contains lines containing one command or file name. The Job file also supports "sections". Currently a "TX" and a "RX" section for files are supported. If no section specified a "TX" section is assumed and all files listed are transmitted. "RX" marks files to be retrieved and/or received*.

* If the optional parameter **H** e.g. **/MsH** is set, the client can't request a specific file but it can receive files automatically sent by the remote server.

Sample job file: job001.txt

```
# Sales man 001
#RX
leads.dat
orders.dat
#TX
sales.dat
```

A *Job File* is a text file and can be created using a simple editor. Every entry must start in the first column. A command begins with '#'. A file name is entered as is. Comments start with '#' followed by a space. An extended version of the above sample including a manual login procedure – the same as the ZMCS server expects follows:

Sample job file: job002.txt

```
# - Manual Login -
#SendStr <13>
#DELAY 1000
#TIMEOUT=5000
#WaitForStr "Username:"        # wait for the prompt
#OnFailure L_END
#SendStr dorian<13>           # send user name
#
#WaitForStr Password:         # wait for the next prompt
#OnFailure L_END
#SendStr secret01<13>         # send password
#WaitForStr Welcome           # wait for the welcome string
#OnFailure L_END
#
# Sales man 001
#RX
leads.dat
orders.dat
#TX
sales.dat
#L_END
```

Commands and Syntax

Commands start with '#' and are expected in the first column of the line e.g.

```
#SendStr Hello World
```

Commands ignore character case. If a command expects parameter it should be placed after a space character or the equality sign.

Parameter containing text can be placed between quotes ("") in order to ensure where text ends (e.g. preserving spaces at end) and to see clearly where it ends if the editor does not show the end of line e.g.

```
#WaitForStr "Username:"
```

Binary data (non printable characters) e.g. ASCII 1 to 31 and 125 to 255) can be transmitted by placing them within '<>' e.g.

```
#SendStr "Hello World<13>"
```

If the real character '<' or '>' is required it should be doubled e.g.

```
#SendStr "Hello World>>"
```

will transfer the string "Hello World>" to the remote station.

Script-Commands

The script commands are forming a small programming language with simple syntax. Actually the following commands are supported:

..

A comment line starts with the special character '#'. A space (' ') or a minus character ('-') can follow but must not. A '#' alone is also valid a comment line.

```
#
#-
#.....
# - Login -
```

Inline comment is also supported:

```
#SendStr "<13>"  # sending a wakeup to the server
```

Empty line

An empty line is valid and will be ignored like a comment line. An empty line is completed, as every other line, with CR (ASCII 13) and/or LF (ASCII 10).

#Lname

A label is a one line marker used for branching within the script. A label starts with the sequence '#L' followed by a unique name. Branching to a label is done by using the commands *#Goto, #OnSuccess, #OnFailure, #OnTrue, #OnFalse*.

```
#L_TXFILES
```

#TIMEOUT=*value*

Defines the timeout value in milliseconds used by commands like *WaitForStr, SendStr, ReceiveStr* nutzen. The default value is 1000 (1 Second).

```
#TIMEOUT=5000
```
Defines 5 Seconds.

#ReceiveStr *[String]*

The command *ReceiveStr* receives data until the specified string is received or until the amount of data as set in *#RXCOUNT* received or until the timeout occurs (see also *#TIMEOUT*) or the internal buffer is completely filled up (2048 bytes/characters). The string parameter is optional. If no parameter specified it will wait and collect data up to *#RXCOUNT,* or timeout occured or the internal buffer is completely filled up.

```
#ReceiveStr
#ShowStr
```
Will wait for a string up to the timeout and then will print it out.

The string to wait for will be checked case insensitive.

#SendStr *String*

The *SendStr* command transmits data.

```
#SendStr "Hello World<13>"
```
Transmits the string "Hello World" followed by a carriage return (ASCII 13) to the remote station.

#WaitForStr *String*

The *WaitForStr* waits and receives data until the specified string is received or until the amount of data as set in *#RXCOUNT* received or until the timeout occurs (see also *#TIMEOUT*) or the internal buffer is completely filled up (2048 bytes/characters).

```
#WaitForStr "Username:"
```
Waits for the string "Username:".

The string to wait for will be checked case insensitive.

#GOTO *Label*

This command directs script to jump to a labelled line. It performs a one-way transfer of control to another script line. The next line to be executed will be the one immediately following the label.

```
#GOTO L_END
```

#ONFAILURE *Label* alias #ONFALSE *Label*

This command is a conditional version of the #GOTO. It checks the 'result' of the last operation (e.g. *IFINSTR, ReceiveStr, SendStr, WaitForStr*) and if it was not successful it jumps to the specified label.

```
#ONFAILURE L_END
```

Translates to: *If LastError<>0 Then Goto L_END*

#ONSUCCESS *Label* alias #ONTRUE *Label*

This command is a conditional version of the #GOTO. It checks the 'result' of the last operation (e.g. *IFINSTR, ReceiveStr, SendStr, WaitForStr*) and if it was successful it jumps to the specified label.

```
#ONSUCCESS L_TXFILES
```

Translates to: *If LastError==0 Then Goto L_TXFILES*

#PRINTSTR *String*

Prints the specified string.

```
#PRINTSTR "Line 101"
```

#SHOWSTR

Prints the internal buffer. The internal buffer can hold up to 2048 bytes/characters and is filled by *WaitForStr, ReceiveStr.*

```
#SHOWSTR
```

#SHOWVAR *Variable[,Variablename,...][,Names]*

Prints the value of the specific variable (e.g. *Protocol, RXCount, StrLen, Settings, VAR_I, VAR_J, VAR_K, VAR_A, VAR_B, VAR_C, LINE*) . The name of the variable is expected as parameter but without the '#'. More than one variable can be specified and should be separated by ' ' or ','.
The special parameter „NAMES“ will print the name of the variable incl. a „=“.

```
#SHOWVAR RXCount
#SHOWVAR VAR_I, VAR_A names
#SHOWVAR Line names    # print the actual script line number
```

#IFINSTR *String*

This command scans the internal buffer for the specified string. This check produces a 'result' that can be evaluated with an '#ON..' command. The internal buffer stores data received by *WaitForStr, ReceiveStr.*
The following receives a string and jumps to the label L_TXFILES, when the specified string was received.

```
#ReceiveStr
#IFINSTR "Welcome"
#ONSuccess L_TXFILES
```

#IFNOTINSTR *String*

This command scans the internal buffer for the specified string. This check produces a 'result' that can be evaluated with an '#ON..' command. The internal buffer stores data received by *WaitForStr, ReceiveStr*.

```
#ReceiveStr "<13>"
#IFNOTINSTR "Welcome"
#ONSuccess L_END
```

Receives a string up to the carriage return and jumps to the label L_END is the specified string was not received.

Translates to: *If Not Found "Welcome" Then Goto L_END*

#DELAY *Value*

Suspends execution for *Value* Millisekunden.

```
#DELAY 1000
```

Execution is suspended for one second.

#RXCOUNT =|==|>|<|<>|*!=*|>=|<= *Value*

The variable *RXCount* controls the behavior of the functions *WaitForStr* and *ReceiveStr*. If *RXCount* is set to a value > 0 the function will receive up to this limit characters or until another condition becomes true(e.g. Timeout). If *RXCount* is set to 0 (default) then other conditions, based on the function, will be observed (2048 characters buffer size, timeout, Search-String) and *RXCount* will be ignored.
More details about the supported operators are presented at `#VAR_x`.
Only positive values are accepted.

```
#RXCOUNT=10
#TIMEOUT=2000
#ReceiveStr
#SHOWSTR
#SHOWVAR RXCount
#STRLEN>=10
#ONTRUE L_END
#PRINTSTR „Less than expected“
```

#STRLEN ==|>|<|<>|*!=*|>=|<= *Value*

The variable *StrLen* provides the amount of characters (bytes) stored within the internal buffer e.g. after *ReceiveStr, WaitForStr*.
The variable *StrLen* can only be queried. More details about the supported operators are presented at `#VAR_x`.

```
#RXCOUNT=10
#TIMEOUT=2000
```

```
#ReceiveStr
#SHOWSTR
#SHOWVAR StrLen
#STRLEN<10
#ONTRUE L_GO
#PRINTSTR „Weniger als erwartet“
```

#VAR_x *=|+=|-=|==|>|<|<>|!=|>=|<=* *Value*

The following variables are supported: *A, B, C, I, J, K*. The variables can be used for loop and counting purposes. Positive and negative values can be used.

```
#VAR_I=-10        assign a negative Value
#VAR_I-=1         decrement the variable by Value
#VAR_I+=2         increment the variable by Value
```

The variables *A, B, C, I, J, K* can be changed and queried.

The following logical operations can be applied on variables that also set the 'result' that can be checked using the #ON.. command:

```
#VAR_A==0         check if equal
#VAR_B<>0         check if different
#VAR_C!=0         check if different
#VAR_I>0          check if greater than
#VAR_J>=0         check if greater or equal than
#VAR_K<0          check if less than
#VAR_I<=0         check if less or equal than
```

The value of a variable can be printed out using #SHOWVAR.

Examples

Increment and print value

```
#VAR_A=1    # set o 1
#VAR_A+=2   # increment by 2
#PRINTSTR "Variable A="
#SHOWVAR   VAR_A
#PRINTSTR "<13><10>------<13><10>"
```

A loop using positive values

```
#VAR_I=0    # set to 0
#L_LOOP1
#Delay 300
#VAR_I+=1   # increment by 1
#SHOWVAR   VAR_I names
#PRINTSTR "<13><10>------<13><10>"
#VAR_I<>10  # compare with 10
#ONTRUE L_LOOP1
```

A loop using negative values

```
#VAR_I=-1      # set to -1
#L_LOOP1
#Delay 300
#VAR_I-=1      # decrement by -1
#SHOWVAR   VAR_I names
#PRINTSTR "<13><10>------<13><10>"
#VAR_I==-10    # compare with -10
#ONFALSE L_LOOP1
```

#FLOW =|== *Value-String*

The variable *Flow* defines the flow control (Handshake) for the serial line.
Possible values are: CTS, DSR, XON and in any combination.
More than one shold be separated with ' ' or ','.
More details about the supported operators are presented at #VAR_x.

```
#FLOW="CTS,DSR"
#SHOWVAR "Flow names"
```

#RTS =|== 0 or 1

The variable *RTS* sets or queries the signal state of the serial UARTs RTS line.
More details about the supported operators are presented at #VAR_x.

#DTR =|== 0 or 1

The variable *DTR* sets or queries the signal state of the serial UARTs DTR line.
More details about the supported operators are presented at #VAR_x.

#CTS == 0 or 1

The variable *CTS* checks the signal state of the serial UARTs CTS line and sets the 'result' that can be can be checked using the #ON.. command.

#DSR == 0 or 1

The variable *DSR* checks the signal state of the serial UARTs DSR line and sets the 'result' that can be can be checked using the #ON.. command.

#DCD == 0 or 1 also #RLSD == 0 or 1

The variable *DCD* checks the signal state of the serial UARTs DCD line and sets the 'result' that can be can be checked using the #ON.. command.

#RI == 0 or 1

The variable *RI* checks the signal state of the serial UARTs RI line and sets the 'result' that can be can be checked using the #ON.. command.

#OPENFILE *Pair-Values-String*

OPENFILE opens or creates a file. The parameter string contains "Name=Value;" entries, separated by ';'. Spaces within the parameter string should be avoided.
The parameter string provides the filename („Name=“) and optional options („Options=“). The filename should contain the complete path or else it will be assumed localy to the calling program / module. The options specify what functions are allowed to execute on the file (Read, Write, Append) and the mode (Text or Binary).
Possible values for*Options* are:

```
Options=[READ | WRITE | APPEND | READ WRITE | READ APPEND] [TEXT | BIN]
```

If nothing is set, Read and Text are assumed. If WRITE without a READ is specified on a existing file, it will be deleted first.

```
#OpenFile "Name=acc001.txt;Options=WRITE TEXT"
#WriteFile "One Line.<10>"
#WriteFile "Another line.<10>"
#WriteFile "DONE THIS.<10>"
#CloseFile
```

Actually only one file can be opened concurrently.
OPENFILE sets the 'result' that can be can be checked using the #ON.. command.

#CLOSEFILE

CLOSEFILE closes the file opened with *#OPENFILE*.
CLOSEFILE sets the 'result' that can be can be checked using the #ON.. command.

#READFILE *[Amount]*

READFILE reads data from a file. The internal buffer is used to store the data and from there *#ShowStr* can print or *#SendStr* can transfer the data.
The parameter *Amount* controls the amount of data to read. If the parameter is not specified, it reads complete lines (if text file) or until the internal buffer is completely filled up (2048 characters/bytes) (if a binary file).
If *Amount* is set on a text file the function may read less data due to the length of a line.

READFILE sets the 'result' that can be can be checked using the #ON.. command.

```
#OPENFILE "Name=acc01.txt;Options=READ BIN;"
```

```
#OnFailure L_END
#L_REPEAT
#ReadFile 32  # read xx chars at a time
#ShowStr
#PRINTSTR "<13><10>"
#OnSuccess L_REPEAT
#CLOSEFILE
```

#WRITEFILE *[String]*

WRITEFILE writes data to a file. If the parameter *String* is not specified, the data from the internal buffer will be used.

```
#ReadStr
#WriteFile            # write the buffer received
#WriteFile "End<10>"  # write the parameter
```

WRITEFILE sets the 'result' that can be can be checked using the #ON.. command.

#SEEKFILE *=Amount [FromStart | FromEnd]*

SEEKFILE moves the file pointer *Amount* positions. The move is relative to the current position or relative to the start of file (*FromStart*) or relative to the end of file (*FromEnd*). If nothing additional is set, it moves relative to the current position.
With text files use only "=0 Start" or "=0 End".
SEEKFILE sets the 'result' that can be can be checked using the #ON.. command.

```
#SeekFile 20 FromStart
```

#CONNECT [=] *Pair-Values-String*

Establishes a connection. The parameter string contains "Name=Value;" entries, separated by ';'. Spaces within the parameter string should be avoided.
The parameter string may provide the following entries:

Serial=COM1; | COM2;| , ..
ComType=RS232|RS422|RS485|TCPIP-CLIENT|TAPI|ISDN;
ConnectAddress=192.168.0.1:9100; | 2223334444; # according to the *ComType*
ConnectTimeout=60; # sets for 60 seconds. It cannot be less than 10 seconds.

and also the others entries available in *#SETTINGS* (for serial lines).

Serial
defines the name of the serial line to use. When this is set, the *ComType* need to be set only if RS485.

ComType
defines the communication type. The actual software license must support this.

ConnectAddress
defines the address to connect to. This value depends on the *ComType*. For a direct serial link (cable only no modem) don't set this entry since no modem there to dial.
For serial lines, ISDN set the phone numer to dial.
For TCP/IP set the IP address and the Port address e.g. `192.168.0.1:9000` or `www.mydomain.com:9100`.

ConnectTimeout
defines the timeout value in seconds for the connection attempt. It cannot be less than 10 seconds. For national calls 60 seconds are fine. For international calls set it to 90.

```
#CONNECT "Serial=COM1;BaudRate=115200;Handshaking=CTS;ConnectAddress=2223456;
          ConnectTimeout=60;"
#OnFailure L_RETRY

#CONNECT "ComType=TCPIP-CLIENT;ConnectAddress=www.mynet.com:9100;
          ConnectTimeout=60;"
#OnFailure L_RETRY
```

#SETTINGS [=] *Pair-Values-String*

Changes the actual communication parameters for the serial line. The parameter string contains "Name=Value;" entries, separated by ';'.
Spaces within the parameter string should be avoided.
The parameter string may provide the following entries:

"BaudRate=50..921600" # according to what the UART hardware offers
"DataBits=5-8"
"StopBits=1 | 2"
"Parity= None | Odd | Even | Mask | Space" # only one value, the first letter satisfies
"Handshaking=" None or any combination of "CTS, DSR, XON"

If one entry is omitted its actual value will be used.

```
#Settings "BaudRate=9600;DataBits=8;StopBits=1;Parity=N;Handshaking=CTS;"
#Settings "BaudRate=115200;"  # change baud rate only
```

#PROTOCOL [=] *String*

Changes the protocol to be used for the next file transfer. Provided the actual software license supports the specified protocol.
The parameter *String* provides the protocol name.

```
#Protocol=ZMODEM | KERMIT | XMODEM | XMODEM-CRC | YMODEM | YMODEM-BATCH | ASCII
```

Script-Examples

(Important: Use only pure text files in ANSI (8-Bit) format.)

#ReceiveStr replaces #WaitFor:

```
#WaitFor "Username:"
#OnFalse L_END

#ReceiveStr "name:"
#IFINSTR "name:"
#OnFalse L_END
```

Send binary data

```
#SendStr "<2>Data<3>"
```
will transmit STX Data ETX

Loop using the variable 'I' as control variable'

```
# Looping up to 10 times to label LOOP1

#VAR_I=0
#L_LOOP1
#Delay 300
#VAR_I+=1          # increment by one
#VAR_I<>10         # check if different
#ONTRUE L_LOOP1    # if true goto start of the loop
#PRINTSTR "Variable I="
#SHOWVAR   VAR_I
#PRINTSTR "<13><10>------<13><10>"
```

Manual Login (similar to what expected by the ZMCS server)

```
#SendStr "<13>"
#TIMEOUT=3000
#WaitForStr "Username:"     # wait for the prompt
#IFINSTR "name:"
#OnFalse L_END
#SendStr "dorian<13>"       # send username
#WaitForStr "Password:"     # wait for the next prompt
#OnFailure L_END
#SendStr "secret01<13>"     # send password
#WaitForStr "Welcome"       # wait for the welcome string
#OnFailure L_END

#L_END
```

Handle file

```
#OPENFILE "Options=APPEND;Name=file.txt;"
#OnFailure L_END
#WriteFile "--One Line.<10>"
#WriteFile "--Another line.<10>"
#WriteFile "--DONE THIS.<10>"
#CLOSEFILE
#L_END
```

Switching Protocol

```
#TX  marks a transmit section
# ------------------------------------------------------------------
#Protocol = Kermit    # switching protocol
"atest00.txt"     # use quoted to protect from accidentically added spaces etc.

#Protocol = Zmodem    # switching protocol
"atest01.txt"

#Protocol = Kermit    # switching protocol

#RX  marks a receive/retrieve section
# ------------------------------------------------------------------
atest11.txt
atest22.txt
atest33.txt
```

Return Codes:

The ZMCS software ends returning a return code. The following return codes define if the requested job was successful or, in case of failure, if a retry is useful.

0 Wrong parameters or user canceled execution (Control-C)
1 OK, files transferred
2 Error, retry may be useful
3 Error, retry is not useful

Error Codes

-301 Invalid port number
-302 File not found
-303 Remote Modem does not answer or is busy
-304 Timeout while transmitting data
-305 Synchronization (Handshake) failed
-306 Modem is not connected or power is off
-307 Login failed
-308 Invalid serial number

The return code can be retrieved by the calling program after the ZMCS software ended. Within a batch file (.BAT, .CMD) ERRORLEVEL returns the return code e.g.:

```
IF ERRORLEVEL 1 GOTO S_OK
ECHO Error Nr: %ERRORLEVEL%
GOTO S_END
:S_OK
ECHO Successful!
:S_END
```

Protocol Error codes

SuperCom return codes are negative. Return codes > 0 were returned from system calls
e.g. 2=File not found.
Wrong protocol settings or a poor connection are the major reasons for protocol errors. The following list shows some protocol errors reported while tracking communications errors:

Timeout occured (-9999)

A timeout occured. The remote station failed to answer within the required time frame. On high latency satelite links the timer may need to be increased. See also Q & A, *How to increase timeouts*.

User cancelation (-9998)

Protocol ended on user request.

Too many retries (-9997)

Link o connection is not stable – perhaps too many retries. Check communication and protocol settings on the remote side.

Canceled by the remote user or protocol (-9996)

The protocol received wrong data (-9994)

Compatibility problem. Check communication and protocol settings on the remote side.

Q & A

Does it support binary files ?
Yes, ZMCS can transmit text files and binary files.

What security features does it offer ?
For security reasons, the Server can be configured to accept files only in the current working directory (see parameter /W). Files to be transmitted or retrieved must be located inside the working directory. Other directories are protected. The server can also be configured to accept only *authenticated users* (see parameter /XL),

How can I connect using a third party software ?
The ZMCS in server mode supports the command /MH, which disables the special handshake used. This way a third party application (i.e. Hyperterminal) can connect to the server. The third party application has to start transmitting the file(s) immediately after connection established. The server will disconnected if nothing received within a time frame of approx. 20 seconds. This limit does not apply to a direct link connection.
What kind of handshake is the server using ?
The description of the handshake the ZMCS server uses is available free of charge to customers in order to enable developers to connect to the server using custom functions.

Can't locate the file just received?
This may happen for some reasons:

1. The file transfer was abborted because of too many errors. Usually in this case the error free part is still on disk. The *ZMODEM protocol* for instance enables to resume the file from this point with the next try. If the abbort was with the first data packet, then no data is saved and files received with zero size are deleted.

2. The server couldn't locate the requested file. Please check the file next to the parameter /F. Als check the servers working directory (/W) if the working directory exists and the requested file is stored there.

3. The file not found by the client. Please check the file next to the parameter /F. Also check the clients working directory (/W) if the working directory exists and the requested file is stored there.

Why does the user login fail ?
When starting the server use the parameter **/XL**.
The user configuration file `ZMCS_usr.ini`, located in the servers directory, should contain the same user information as specified with the parameter /XLU=*username*,P=*password* when starting the client.
E.g.
Server
`zmcs.exe /C2 /Ms /XL` (starts the server on COM2 or ttyS1 expecting clients to login)

Client

```
zmcs.exe /C1 /Mt /T123456789 /F@filelist.txt /XLU=dorian,P=secret01
```

(dials and connects on server, performs the login, and runs the filelist)

How to end a server using a program ?

Assume that the server is running on COM3 (ttyS2) and it has to terminate from within a third party application. We use the ZMCS program and the parameter /Mu plus the communication number.

```
 Unloading the server on COM3 (ttyS2): ZMCS /Mu3
```

How to start ZMCS server with starthidden ?

Enter "starthidden" in front of a regular command line e.g.

```
 starthidden zmcs.exe /c3 /Ms
```

How to increase timeouts for ZMODEM ?

Create a SuperCom configuration file called „supercom.ini“ and add the following under the comm port section e.g. COM3 (ttyS2). The following key value define milli seconds.

```
[COM3]
; defaults to 1000 ms
STD_WAITONECHAR=6000
; defaults to 5000 ms
STD_WAITHEADER=12000
```

The default values within the ZMODEM protocol module are

```
STD_WAITONECHAR=1000
STD_WAITHEADER=5000
```

How to increase connect timeouts ?

The connect attempt run up to 90 seconds. This is sufficient for even international calls. If this need to be customized, add the key CONNWAIT in the configuration file „zmcs.ini“ under section „CONFIG“. The key value defines seconds.

```
[CONFIG]
; defaults to 90 seconds
CONNWAIT=120
```

How to increase the timeout value when no handshaking used ?

After connected the third party client must start the ZMODEM protocol within 20 seconds. This timeout value can be adjusted by setting the key TO_SYN_NH within „zmcs.ini“. The key value defines seconds.

```
[CONFIG]
; defaults to 20 seconds
TO_SYN_NH=60
```

Where should be the configuration files stored ?
The configuration files e.g. **zmcs.ini, supercom.ini** are expected to be within the same directory as the application file „zmcs.exe“.
In order to create or edit configuration files use notepad or any other editor and store the file as ANSI (8-bit).

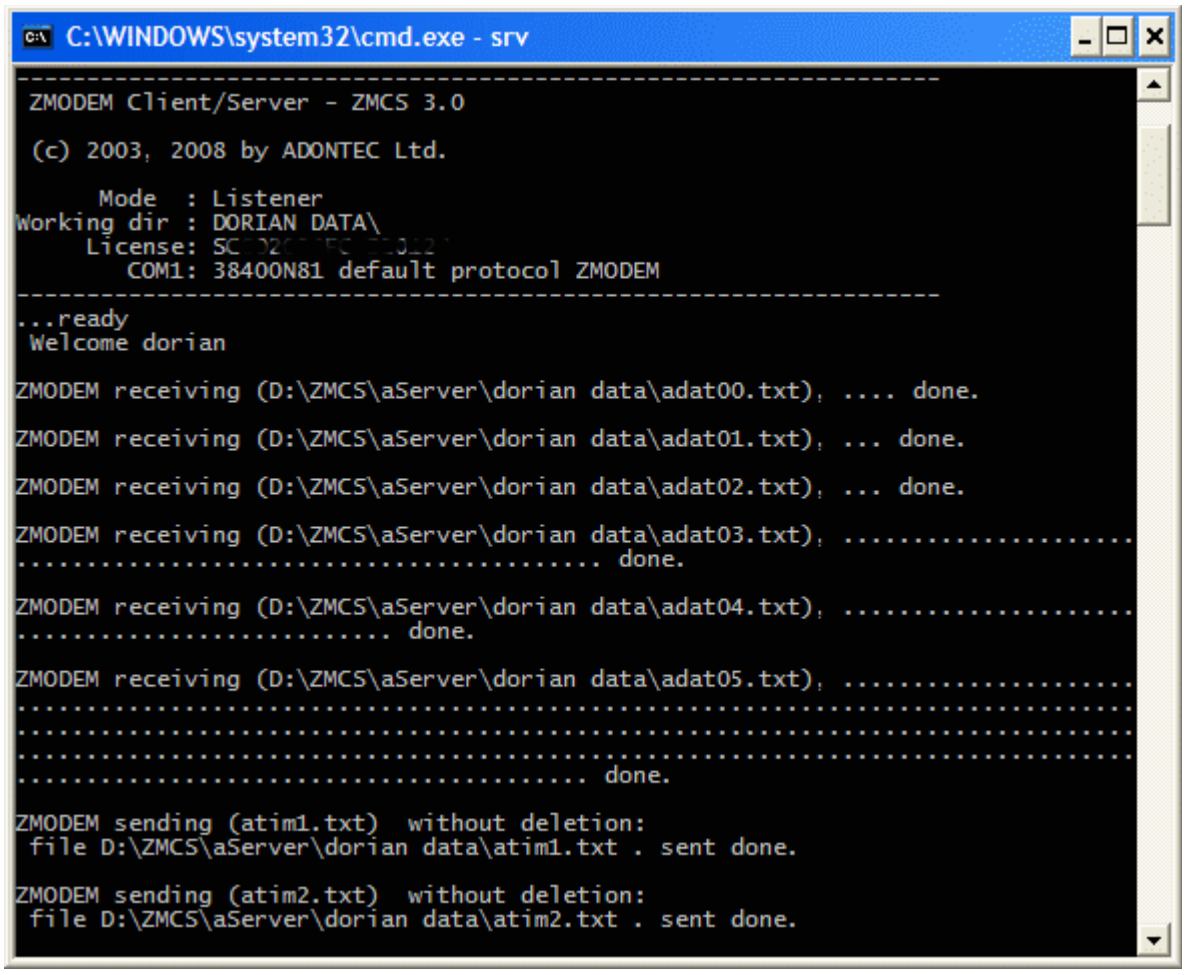
Is it possible to grab the ZMCS output directly into my application ?
Yes. The easiest way doing this is using a DOS pipe "<" redirecting to a file or to your application directly. ZMCS also supports IPC (Interprocess Communication) using the parameter **/XIc**. A small IPC sample program showing this is also available in C/C++ source code.

Can I execute it under Windows 64-Bit (x64) ?
Yes, ZMCS for Windows is as a real 32 Bit Windows application and created also to be compatible for a x64 Windows system. So it can safely be used under Windows x64.

Sample Session

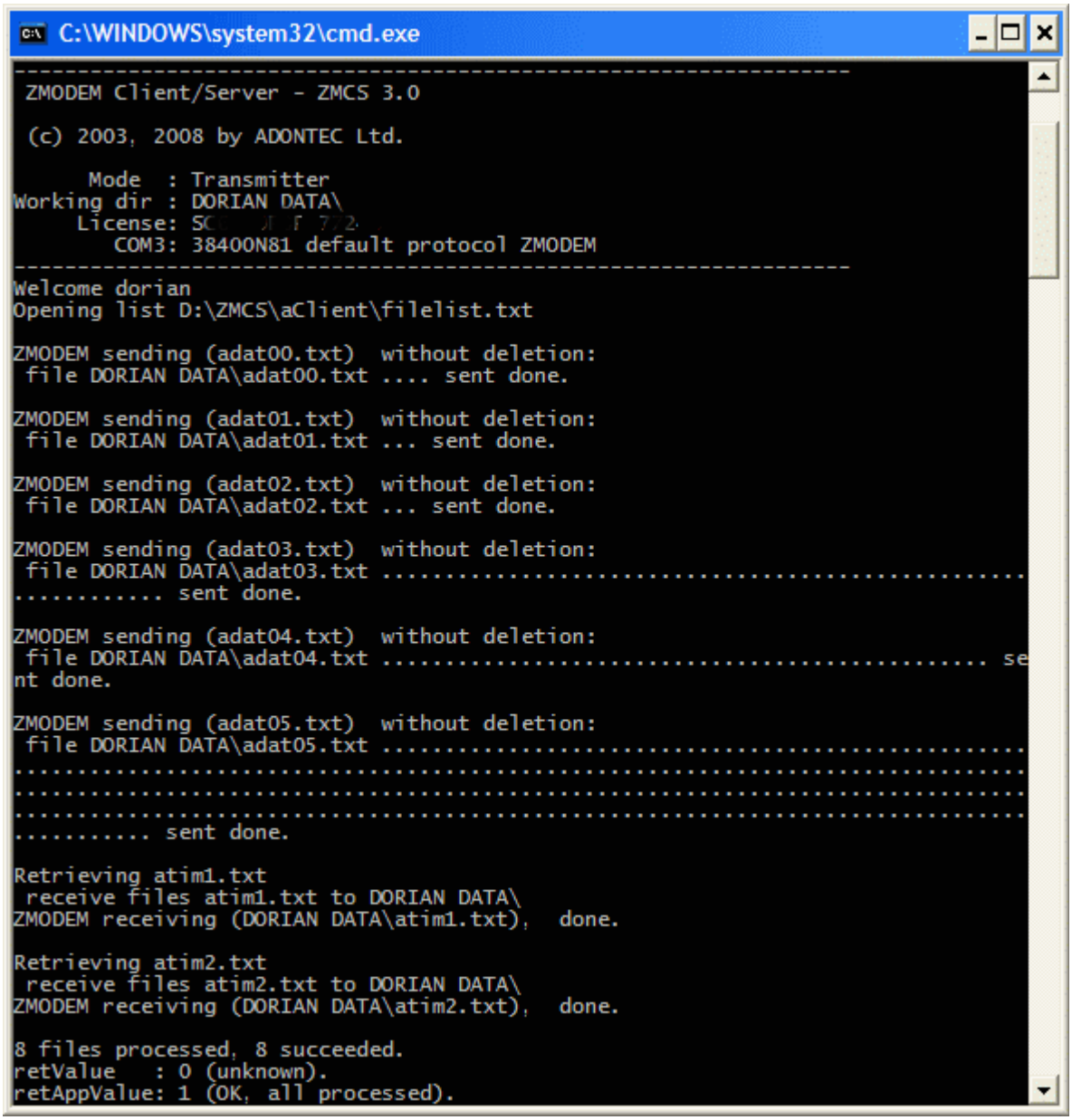
Server

```
zmcs.exe /c3 /ms /L /XL
```



Client

```
zmcs.exe /c4 /mt /L /w"D:\ZMCS\aClient\dorian data" /f@filelist.txt
/XLU=dorian,P=secret01
```



Content of `filelist.txt`

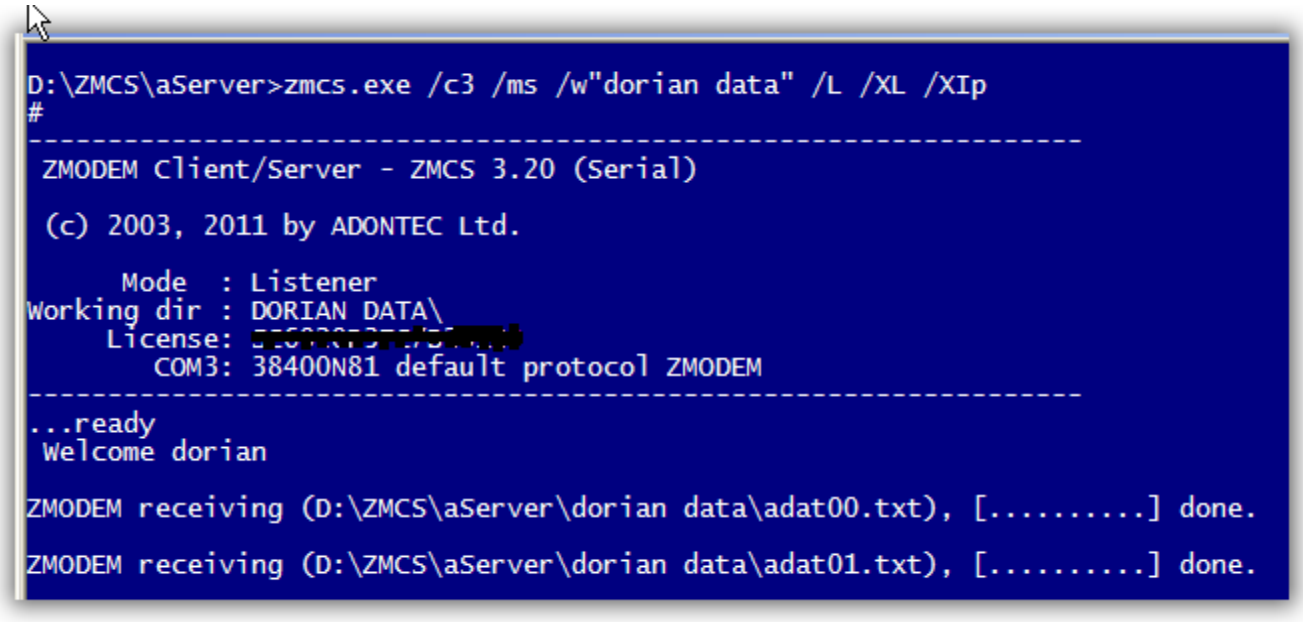
Sample filelist
RX and TX section supported to perform within a connection session.
If no section marked, TX is taken as default
Any line starting with # is a comment line

```
# ---------------------------------------------------------------
#TX  marks a transmit section
adat00.txt
adat01.txt
adat02.txt
adat03.txt
adat04.txt
adat05.txt
#RX  marks a receive/retrieve section
atim1.txt
atim2.txt
```

Sample Session 2

Server

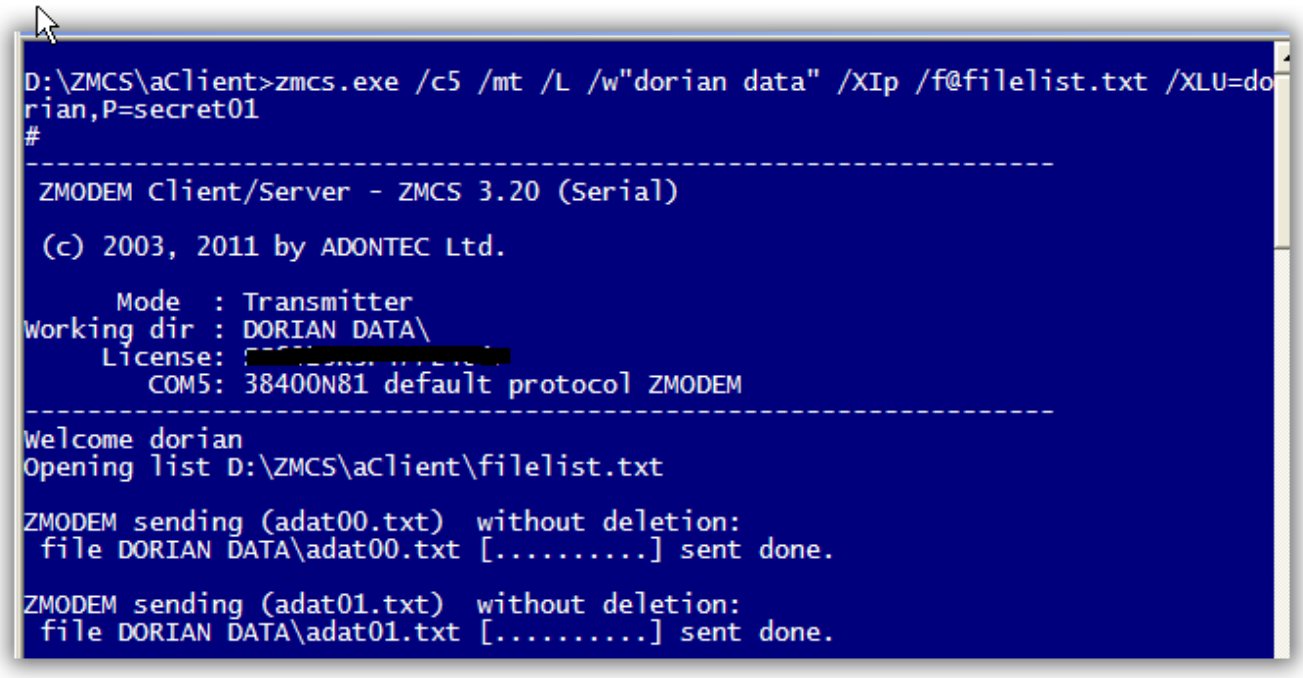
```
zmcs.exe /c3 /ms /L /XL /XIp
```



Progress information shown in percent.

Client

```
zmcs.exe /c4 /mt /L /w"D:\ZMCS\aClient\dorian data" /f@filelist.txt
/XLU=dorian,P=secret01 /Xip
```



Progress information shown in percent.

Configuration

Using Modem

The software should be set to use baud rates at high level. Today modern modems accept 115200 bps and can work with 57600bps or even 38400bps. Setting low communication speed may result to timeout errors.

Line signals

The cable used to connect an external modem should be well equipped supporting the lines: RX, TX, RTS, CTS, DTR, DSR, DCD (RLSD), RI.

Black list

If a modem fails, after a series of dial attempts, to connect to a specific phone numer this number gets black listed. When this occurs the modem does no more dial this number. To overcome this switch the power off for about 3 to 5 seconds and then switch power on.

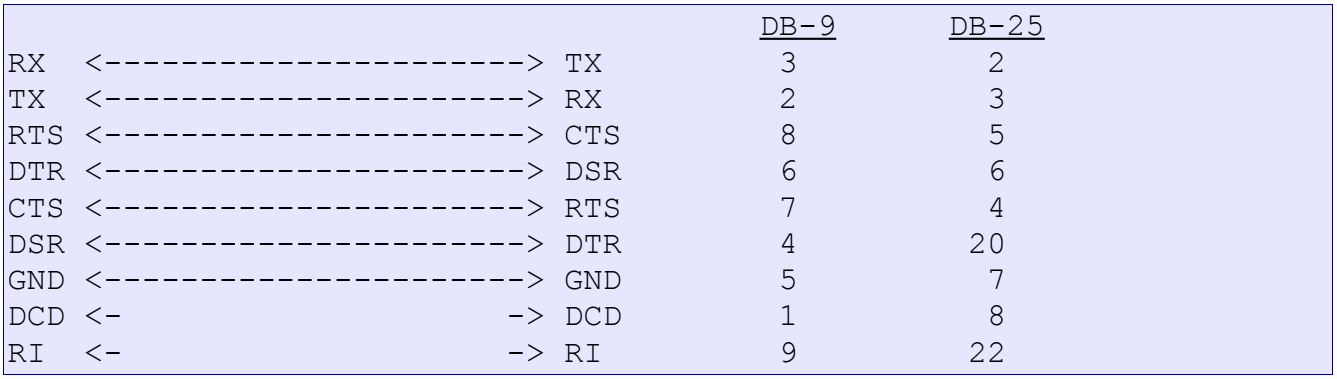
Redial limits

A modem may be pre-configured to allow a redial only after a specific delay. This may be true for specific regions of the earth. This delay may be around one minute.

Using direct link

Serial cable

The cable used to connect the two computers must support the lines: RX, TX, RTS, CTS, DTR, DSR. The cable must provide the remote signal of CTS and DSR and should not be short wired on the local side. RI and DCD signals should be short wired on both ends. Fully equipped "NULL-Modem" cable do that.



DCD can be short wired locally with DSR. In order to avoid false interrupts it is a good idea to short wire RI with the DSR or DTR line locally.

Installation

Please follow the directions below in order to install this software.

You are allowed to install or execute the same amount of copies as you own serial numbers.

Windows

It only needs the following files be copied into the applications directory.

- zmcs.exe
- lic.txt <-- *put your serial in here*
  - supercom.dll (if included)
  - protocol.dll (if included)

Linux

It only needs the following file be copied into the applications directory.

- zmcs
- lic.txt <-- *put your serial in here*

DOS

It only needs the following file be copied into the applications directory.

- zmcs.exe
- lic.txt <-- *put your serial in here*

Licensing

Please keep in mind that every ZMCS you install requires it's own unique serial number.
(Read further on next page)

Installation and Distribution

The file **lic.txt** must be included with your installation. The first line must include a unique serial number. Each serial number is allowed to be used only one time. You need one license for every PC you want to install this software on.

If you need more licenses please contact our sales staff to receive a quote for discounted multi-license software packages.

If you use more than one serial number (e.g. Serial and TCP/IP) you can add additional lines
e.g.

```
SN:SC1234serial567
SN:SC1234tcpip567

(c) ADONTEC LTD. All rights reserved.
www.adontec.com
```

Bitte make sure you close each line by pressing [ENTER] key.