

RSU 1.5

Remote Software Update for Networks—Documentation

Copyright © 1992-1995 Burkhard Daniel & Hans-Georg Michna

Table of Contents

| | |
|--|--|
| Shareware Licensing and Registration..... | |
| Support..... | |
| Disclaimer..... | |
| Introduction..... | |
| How RSU Works..... | |
| Installing RSU..... | |
| RSU Server Preparation..... | |
| Workstation Preparation..... | |
| The RSU Development Cycle..... | |
| Steps in the RSU Development Cycle..... | |
| Prepare an RSU Test Workstation..... | |
| Create and Test the New Module..... | |
| Protect Users From Your Tests..... | |
| Upload the New Module to the RSU Server..... | |
| Adapt the RSU Program..... | |
| Test the New RSU Program..... | |
| Activate the New RSU Program..... | |
| Programming RSU..... | |
| General Rules..... | |
| Sample RSU Program..... | |
| Execution Sequence..... | |
| Alphabetical List of Commands..... | |
| General Command Syntax and Semantics..... | |
| Sections..... | |
| Environment Variables..... | |
| DOS Commands..... | |
| Echo..... | |
| Goto..... | |
| If..... | |
| IniAddLine..... | |
| IniChangeLine..... | |
| IniCopyLine..... | |
| IniCopySection..... | |
| IniDeleteLine..... | |
| IniDeleteSection..... | |
| SynchronizeDir..... | |
| SynchronizeDir (Old Syntax)..... | |
| Brief Command Overview..... | |

Shareware Licensing and Registration

RSU including its documentation files is Copyright © 1992-1995 Burkhard Daniel and Hans-Georg Michna. It is distributed under the Shareware concept. Its use on isolated PCs, for example to change settings in WIN.INI through batch files without the involvement of any network file server is free, even if the computer is connected to a network. As soon as RSU is used to copy files from a network file server to a user's local storage or to copy any parts of files of the Windows INI type from a network file server, however, its free use is restricted to a 30 day trial period. After that the shareware fee has to be paid, if the program is still used.

Since the combinations of file servers and workstations can be rather involved, some specifications are in order. If RSU is used to manipulate shared files on other file servers (for example several slave servers being updated from one master server), then each target file server being updated counts only as one user, as long as no individual workstation files are modified on it. In other words, user workstations do not count if RSU is not used to alter their individual files like CONFIG.SYS, AUTOEXEC.BAT, WIN.INI, SYSTEM.INI, PROGMAN.INI, PROTOCOL.INI or NET.CFG. A special case is that each workstation has its individual files in a workstation specific directory on a file server and RSU is used to update the workstations' individual files on that file server (example: diskless workstations). In this case, each workstation should be counted as one user, since RSU is doing exactly the same amount of work as if the workstation specific files were on local workstation disks.

The shareware fee for RSU 1.x is US \$59.00 or DM 85.00 (German Mark) for each group of up to 100 users. In other words, if the program is used for 1 to 100 users, the fee is US \$59.00 or DM 85.00. If it is used for 101 to 200 users, the fee is US \$118.00 or DM 170.00, for 201 to 300 users US \$177.00 or DM 255.00 and so on. Prices may change in future versions if the exchange rate changes considerably. Future enhanced versions of RSU may be more expensive.

In Germany, add the legally prescribed Value Added Tax (currently 15%).

The program contains no technical means to enforce payment other than noting the requirement on screen when an unregistered copy of RSU is started.

To pay the fee through CompuServe log on to CompuServe and enter the command: GO SWREG. Search for the keyword RSU and register according to the choices you get on screen. Each single registration or license is good for up to 100 users.

Alternatively, send a check and your e-mail address (or fax number) to one of the following addresses.

USA: A.C.I. Micro, Mark Jordan, 45 Roland, Winchester, MO 63021

World: A.C.I. Micro, Hans-Georg Michna, Notingerweg 42, D-85521 Ottobrunn

In Europe use a Eurocheque for DM 85 or equivalent for each 100 users.

In Germany add the legally prescribed Value Added Tax (currently 15%).

If you require an invoice, please ask for it by e-mail or send a purchase order to A.C.I. Micro in Germany. But please keep bureaucracy to a minimum. If you can, please register directly through CompuServe's SWREG service, which is much more convenient and probably cheaper for you as well as for us. Within the European community please add your VAT ID. The VAT ID of ACI Micro is: DE 129 2759 98

This is a sample SWREG session in CompuServe:

```

!go cis:swreg <---{User entry after exclamation mark}
Shareware Registration SWREG

1 Instructions to Register Shareware
2 Register Shareware
3 Instructions to Submit Shareware
4 Submit Shareware (Authors)
5 Shareware Beta Forum +
6 ASP/Shareware Forum +
7 Provide Feedback
8 Frequently Asked Questions
!2 <---{User entry after exclamation mark}

...

Press <CR> to continue! <---{User entry (return key) after exclamation mark}

Please select the geographic region
for which you will be registering shareware.

1 United States
2 Canada/Mexico
3 Europe
4 Asia/Pacific Rim
5 Central/South America
6 Africa
7 Middle East
8 Australia/New Zealand

Enter region number: 1 <---{User entry after colon}
Register Shareware

SEARCH BY:

1 Registration ID
2 Title
3 File Name
4 Author's User ID
5 Author's Name
6 Keywords (Categories)

Enter choice!6 <---{User entry after exclamation mark}
Register Shareware

Enter Keyword (ex. - GAMES): rsu

Reg ID: 552                      Fee (US$): 59.00
                                Shipping/Handling (US$): 0.00

Title: RSU Version 1

File Name: RSUX.EXE              Author: Hans-Georg Michna [74776,2361]
Size: 69200                      Compression: LHA
Computer Type/Operating Systems: DOS, WINDOWS, OS/2

Support: CompuServe Mail to 74776,2361

Forum: GO NOVLIB                 Library: Shareware                 Library Number: 15

RSU V1.0 (Remote Software Update for DOS) is a program that can
- determine whether a user has the latest update level,
- copy, delete, add lines or whole sections from one INI file to another,
- add multiple similar lines (like several device= lines in SYSTEM.INI),
- insert environment variables,
- synchronize directories, i.e. make them equal without unnecessary copying
and more. Shareware $50 for 100 users. Upload by author.

Would You Like to Register? (Y/N) y <---{User entry after "(Y/N)"}

```

When you license RSU, you will receive information including hints on how to upgrade and a special key to turn RSU into a registered version, but you will not normally receive any paper mail. Neither

will you receive the program itself, which is available only on CompuServe and on the Internet. You should retrieve updates from wherever you received the program in the first place. From time to time you may receive information on upgrades and related programs.

The program may be freely distributed as long as the files stay together unaltered and packed in one file using an archiving program like PKZIP, LHA or similar. It is not allowed to add any other files other than minor additions that do not exceed the size of the original files. It is not allowed to distribute RSU as part of another program or package because we fear that this might look as if RSU may no longer require its shareware fee payment which it always does according to the rules outlined above.

We have in the past sent information by e-mail when there were significant changes to RSU. But you have to download the new versions on your own. You don't have to pay anything again for any upgrades to RSU version 1.x. Should there be a significant technological change, like a new operating system, like a 32 bit OS, then there might be an entirely new version of RSU, which will be handled differently, probably as an entirely separate product.

Support

Support is available via e-mail from Hans-Georg Michna 74776,2361 and from Burkhard Daniel 100342,2050. This support is free. In our experience, luckily, RSU needs very little support, because it doesn't seem to cause many problems. But if you need help, usually in the beginning, please feel free to send e-mail. The same holds for questions, remarks or proposals for future enhancements of RSU. You can also reach us through Internet mail: 74776.2361@compuserve.com and 100342.2050@compuserve.com (Please note the periods in the place of the commas).

Disclaimer

At the present state of software technology it is not possible to create error-free software. RSU may contain errors. Anybody using it should take precautions like creating safety backups in case a defect causes damage to any computer or data. The authors of RSU can under no circumstances be held responsible for any damage.

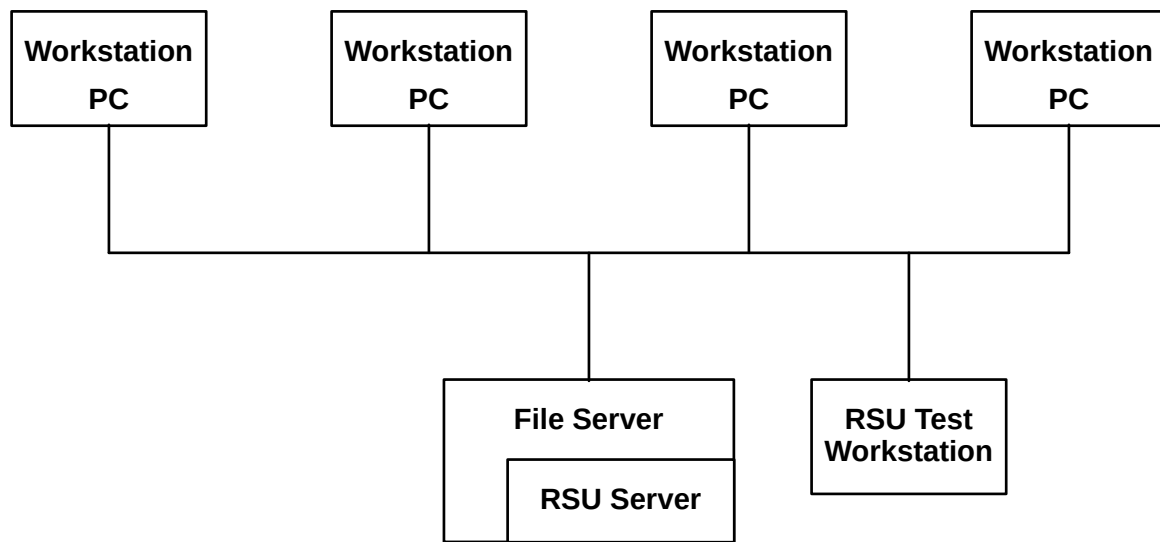
Microsoft® is a trademark of Microsoft, Inc.

Introduction

RSU (Remote Software Update) is a program that updates the software on many individual workstation PCs connected to a file server. The file server, when equipped with RSU, becomes an RSU server.

The RSU program does not run on the RSU server, however. It runs only on the workstations. The RSU server consists only of subdirectories and files on the file server.

¹ The term "workstation" is used here for all user PC's connected to the network, using DOS, not for engineering workstations running Unix or the like.



Sample RSU installation

The fundamental idea is that all workstation configurations are stored on the RSU server. RSU then copies the appropriate modules to each workstation while retaining individual settings or individual software on the workstations.

The file server used for RSU does not have to be the same file server that is used for normal work. It is necessary, of course, that users log in to the RSU server from time to time to get the topical remote software update.

Since the file server does not usually run DOS a separate RSU test workstation is needed to develop and test the configurations. If testing on different hardware is needed then one RSU test workstation is needed for each hardware setup. RSU test workstations can be used for other work when they are not being used for RSU development. Their RSU related configuration has to be totally reset, however, before it is used for testing. RSU can

- 1 copy predetermined files from the RSU server to the workstations,
- 2 change the contents of subdirectories, even whole subdirectory trees, on the workstation PCs such that they become equal to their parents on the RSU server, by adding, deleting or overwriting files on the target PC,
- 3 exclude certain subdirectories and files from directory replication,
- 4 report all changes when synchronizing directories,
- 5 add, change, copy or delete certain sections or certain lines in INI files like WIN.INI and SYSTEM.INI,
- 6 copy or delete sections in text files like AUTOEXEC.BAT and CONFIG.SYS,
- 7 find out whether each workstation is already up to date and skip the updating process,
- 8 detect BIOS signatures of certain hardware and, for example, automatically install the appropriate drivers.

How RSU Works

Since the file server or any program running on any workstation has normally no access to any local disks the main RSU program has to run on each workstation to work its magic. The most convenient way to achieve this is to run it each time any user logs in to the RSU server.²

This can be achieved in different ways on different operating systems. On a Novell network, for example, a command can be called up after the system login script finishes, by using the Novell EXIT "<command>" syntax or by calling it in the middle of the login script with the # syntax. See the manual of your network for details. RSU can be called from a DOS batch file. It has one command line parameter which is the path of the RSU program file. Example:

```
f:\rsu\rsu.exe f:\rsu\rsu.prg
```

RSU.EXE will execute the RSU program file. Files are typically copied from the RSU server to the local disk or modified on the local disk. Instead of the local disk the individual user storage space may also be on a file server. In this case each user's private storage should be mapped to a drive like U:, such that each user sees his private storage area when he refers to U:.

Installing RSU

RSU Server Preparation

Each file server can be a RSU server. The basic method is to keep a mirror image of a workstation in a RSU server directory, for example in F:\RSU\WSIMAGE. There can be several such directories for several different configurations. The configurations have to be created and tested on test workstations, then copied to the RSU server. In addition the RSU server needs one other directory with read-only access for RSU.EXE and the RSU program file (example: RSU.PRG, example of RSU read-only directory: F:\RSU).

The management of more than one workstation configuration can easily be achieved by creating small signal files on the different workstations which indicate the configuration type. The existence of any of these signal files can then be queried by a simple "If Exist" command. Another method is the BIOS scan, using RSU's BIOS function. Thus the BIOS version of the computer or, for example, of the display adapter can be determined automatically and the appropriate configuration installed without any manual intervention.

Workstation Preparation

Before installing RSU you have to make sure that all workstations connected to the RSU server have a valid minimal installation. The absolute minimum would be an installed DOS and the minimal network drivers to be able to connect to the server.

It is recommended that you have an initial workstation setup procedure to erase all RSU related directories and files from a workstation and recreate the whole minimal setup from scratch. This could be done with a batch file like INSTALL.BAT on the RSU server. This will be very useful for users if

² Another method is to run it each time the workstation is started, from the AUTOEXEC.BAT file. Since users have few rights on the file server at that time it would be necessary to adjust those rights such that all users have reading rights to all RSU data without being logged in. This may or may not be possible on different network operating systems.

they have somehow destroyed vital files on their workstation. They will then have a way to get up and running again if everything else fails and no service person is available.

It is also conceivable to use RSU to such an extent that each and every file on the workstation is covered by RSU. This would have the advantage that every workstation would recover from any loss or mutilation of files automatically when logging on to the RSU server. But this will often not be practicable for performance or other reasons.

RSU will run inside Windows and Windows NT. Note, however, that Windows or Windows NT may keep certain files open, such that these files themselves cannot be copied or updated. On Windows NT, RSU, like many other DOS programs, requires read and write access to the file CMOS.RAM in the SYSTEM32 directory. RSU, like any DOS or 16 bit Windows program, cannot use long file names, but it will use the automatic substitutes generated by Windows NT.

The RSU Development Cycle

Steps in the RSU Development Cycle

The RSU development cycle is the process of developing a new configuration. Frequently this will just be a small change in an existing configuration, but it could also be an entirely new configuration for a new type of workstation, for example. Development proceeds in these steps:

1. Prepare a RSU test workstation by making it equivalent to the topical RSU update level.
2. Create and test a new configuration or modify the existing one on a test workstation.
3. Temporarily protect users from your RSU server changes.
4. Upload (copy) the new workstation configuration to the RSU server or modify the existing one on the RSU server.
5. If necessary, adapt the RSU program file (example: RSU.PRG).
6. Test the new RSU setup on a test workstation.
7. Activate the new setup, such that it gets installed on all target workstation PCs (undo step 3).

In many cases this development cycle can be shortened by skipping certain actions if their effect is minimal or if the number of workstations is low enough such that some risk can be taken.

The following sections describe these actions in detail.

Prepare an RSU Test Workstation

First you have to make sure that your RSU test workstation is equal to a topical workstation elsewhere in the network. If the test workstation has not been used for anything that might have altered the files and directories concerned then it can be used right away. If not, and whenever there are any doubts, the test workstation should be installed fresh from the RSU server. It is a good precaution to log in once after installation and obtain the latest changes from the RSU server to make sure they are included in the workstation image on the RSU server.

The RSU update level information can be stored in any file for each workstation. You only have to edit this file or touch it such that the file date or time is changed to force an update.

Start the RSU program, normally by logging in to the RSU server. The test workstation should automatically be updated to the topical update level. After the test workstation is up to the present standard you can now make the desired changes.

Create and Test the New Module

Now you can make the desired modification on the test machine only. Test thoroughly, since all your users who use that module will depend on your conscientiousness.

Protect Users From Your Tests

As soon as you touch the RSU server all users who happen to use the server will receive any modification. Therefore you have to make sure that this does not happen.

There are several ways to protect users:

8. Deactivate the whole RSU server until you are done with all changes. One way is to rename the RSU program. Without it RSU will not be able to do anything. Another is to activate a jump over the RSU part of the controlling batch file.
9. Leave the RSU server running and create a second RSU server at least for the module you intend to work on. This is not quite so difficult as it sounds. It may be necessary for bigger changes that require some time to work out.
10. If you are making a very small change and you are absolutely sure that even a mistake can do no harm you may risk to work on the hot system. But be careful! Depending on the number of workstations and likelihood of a login you should be able to make the change within seconds. This might be viable if you just want to change a few files, for example. Again do not forget to change the date and time of the RSU update level file afterwards, so all users get a new update when they log in.

Upload the New Module to the RSU Server

After you have tested the new configuration thoroughly on your test machine you can now copy the modifications to the RSU server (for example to the F:\RSU\WSIMAGE directory and its subdirectories). It is useful to have a program that can detect the difference between files on two drives like the Norton Commander™ with its “Compare directories” command. In any case be sure to copy all changes from the test machine to the RSU server.

Adapt the RSU Program

Now change the RSU program (example: F:\RSU\RSU.PRG) if necessary. The modified RSU program should be able to copy all modifications from the RSU server area to any workstation PC.

If you make changes to files by means of direct manipulation by the RSU program, for example with IniChangeLine, be sure to make those changes on the original files on the RSU server as well, such that users who do an install from scratch also get them immediately before they receive the next RSU update.

Test the New RSU Program

After the update is entirely in place but not yet activated you should test it before releasing it to all users. For this you either need a second test machine or you have to reset the first one to the previous configuration which is still standard for all other users.

Then you have to make sure that the new update affects only the test machine but not yet all the other users. There are several ways to achieve this. The easiest is to modify the RSU program file such that only the testing user gets the update. Example:

```
...
If %USER% <> SUPERVISOR Then
    Goto Not_yet
End If
rem   Here enter the RSU commands to be tested.
Not_yet:
...
```

This sample batch file fragment presumes that the network user name was written into the environment variable USER, for example with the Novell Netware login script command:

```
DOS SET USER="%USER_ID".
```

Check whether the update is correct. You may have to test each configuration separately if there are several.

Activate the New RSU Program

Finally, after you have convinced yourself that the new update works correctly, you can release it to all users by removing any blocking commands you might have inserted. From that very moment all users who log on will receive the update.

Programming RSU

General Rules

RSU is an interpreter for the RSU control language which strongly resembles BASIC and, to some extent, DOS batch files. The RSU program is the file which controls all RSU operations. It should reside on the RSU server, for example as F:\RSU\RSU.PRG. All users should have read-only access to it.

RSU is started from DOS with either of the following commands:

```
RSU <program file name>
RSU.EXE <program file name>
RSU <program file name> /debug
RSU.EXE <program file name> /debug
```

The /debug switch produces a display of all commands, as they are executed.

As long as the program is unregistered, it always displays a shareware registration screen first. Apart from this, however, the program is fully functional and may be tested with all functions for 30 days. After that time the program must be registered for further legal use, although the program would technically still function properly.

Sample RSU Program

This is an example of an RSU.PRG file:

```
rem    RSU.PRG file

rem    First determine whether the user already has the latest version:

If c:\rsu\version.txt Equal f:\rsu\version.txt Then
    echo Your workstation has the latest update level.
    Goto Finish
End If

rem    Let user know what's going to happen to his computer:

type f:\rsu\version.txt
pause

rem    Modifications to WIN.INI:

IniCopySection f:\rsu\wsimage\win31\win.ini c:\win31\win.ini [fonts]
IniCopySection f:\rsu\wsimage\win31\win.ini c:\win31\win.ini [devices]
IniDeleteSection c:\win31\win.ini [ObscureOldProgram]
IniCopyLine f:\rsu\wsimage\win31\win.ini c:\win31\win.ini [windows] load
IniDeleteLine c:\win31\win.ini [fonts] Swiss=
IniChangeLine c:\win31\win.ini [mail] mailbox=%USER%

rem    Modifications to SYSTEM.INI:

IniAddLine c:\win31\system.ini [display] svgamode=98

rem    Synchronization of user files:

SynchronizeDir
    From f:\rsu\wsimage\win31
    To    c:\win31
    Add
End SynchronizeDir

SynchronizeDir
    From f:\rsu\wsimage\util
    To    c:\util
    Add
    Overwrite
    Delete
    Subdirectories
End SynchronizeDir

rem    Users with HappyVGA adapters get a new driver and support
rem    utilities. This requires that such users have a signal file
rem    c:\rsu\happyvga.sig:

If Exist c:\rsu\happyvga.sig Then
    SynchronizeDir
        From f:\rsu\wsimage\happyvga
        To    c:\happyvga
        Add
        Overwrite
        Delete
        Subdirectories
    End SynchronizeDir
End If

rem    Now install version text file to make sure that this
rem    user doesn't get the same update again:

copy f:\rsu\version.txt c:\rsu

Finish:

rem    End Of File
```

Execution Sequence

RSU is a pure and simple interpreter. This has consequences especially when using the Goto command in connection with If, Then, Else constructs. RSU takes these commands exactly in the sequence they are processed. This means that a Goto jump into an If structure can cause unexpected results if the programmer isn't aware of the actual processing sequence. Therefore it is not advisable to jump into an If command. Jumping out of an If command doesn't hurt. The If construct is simply never completed. But if it is done many times, for example in a loop, it will eventually cause a memory overflow. You should therefore try to use the Goto command sparingly and prudently. Ghastly mistakes can happen if the interpreter encounters, for example, an Else command after jumping out of an If construct, because the interpreter would assume that the Else belongs to the last encountered If command.

Alphabetical List of Commands

General Command Syntax and Semantics

All control files are text files in which each line is followed by a carriage return/line feed pair (13_{dec} and 10_{dec}).

Spaces and tab characters in the beginning of any line are totally ignored. In effect it is as if those characters were removed before executing the RSU.PRg program.

Lines beginning with "REM " or ";" (a semicolon) are treated as comments and not otherwise executed.

Upper and lower case are functionally equivalent. However, the case is retained when information is forwarded into another file.

Command parameters are separated by spaces, with the exception of section headers (section name in brackets) and INI lines after a section header. If a parameter itself contains one or more spaces, it has to be enclosed in double quotes (character no. 34 in the ASCII/ANSI alphabet), so it is not mistaken for several separate parameters. If a quote character itself is to be included, two adjacent quote characters have to be written, but this is possible only within another pair of quotes.

Examples:

```
If Bios(C000-C080) = "ATI GRAPHICS" Then
    ...
End If

If %COMPUTER_TYPE% = "Label ""Taiwan""" Then
    ...
End If
```

The second example compares the environment variable COMPUTER_TYPE with the following text, including the two quotes: Label "Taiwan"

If the first word in any line is a valid RSU command then it is executed. If not the line is deemed to be a DOS command and is forwarded to the DOS command processor. Note, however, that not every DOS command can be used. For example, the DOS command SET has no effect because it is running under a secondary command processor that has no access to the primary environment.

In the syntax definitions below, a few special characters and words are used that have a special meaning.

A word enclosed in angle brackets is a placeholder for a user defined, variable entry:

< >

The following placeholder stands for any valid RSU command:

```
<command>
```

An ellipsis stands for “more of the same”:

```
...
```

Braces and vertical bars are used to define a user choice of one out of several units. Only one can and must be chosen:

```
{ <choice 1> | <choice 2> | <choice 3> }
```

Sections

Several commands work on sections in .INI files. A section is recognized by its header. It ends before the next section header. A section header consists of a section name in brackets. Examples:

```
[windows]
[HPPECL5A,LPT1:]
[Microsoft Word]
```

The section header must begin in column 1, i.e. in the leftmost position of a line. Technically spoken, the opening bracket must immediately follow the carriage return, line feed pair that ends the previous line, or it must be the first byte of the whole file. All lines following the section header belong to this section until another section header follows.

Note that section headers can contain spaces. RSU still recognizes the section header by the enclosing brackets. You don't need double quotes when referring to a section header in an RSU command.

To facilitate manipulation of sections in files like AUTOEXEC.BAT or CONFIG.SYS, a second, alternative form of section header is also recognized, which consists of the abbreviation REM in upper, lower or mixed case, followed by exactly one space, followed by a section header as described above. The R in REM must be in the leftmost column. Examples:

```
REM [drives]
Rem [NETWORK DRIVERS]
rem [User Specific Settings]
```

Warning: Never use the alternative REM syntax inside an RSU command file. All RSU commands work without containing the word REM, even if the target files contain it.

Hint: If you want to turn such REM lines into real comments rather than RSU section headers, insert at least a second space or any other characters between REM and [.

Hint: While all RSU commands will recognize and accept this alternative syntax and work on it and in the sections thus designated, only the command IniCopySection can put such a section header into a file.

Environment Variables

Environment variables can be inserted in any command with the syntax:

```
%<environment variable>%
```

Example:

```
IniChangeLine C:\WIN31\WIN.INI [mail] mailbox=%USER%
```

This example will take the name from the USER= entry in the environment and substitute it for %USER% before executing the IniChangeLine command. For example, if the environment contains an entry reading:

```
USER=DANIEL
```

then the command above will be changed to:

```
IniChangeLine C:\WIN31\WIN.INI [mail] mailbox=DANIEL
```

which is useful for example to save users the separate logging in to Microsoft® Mail.

The substitution happens before any quote characters are processed, i.e. environment variables can be used inside quotes. To use a percent character (%) for other purposes, you have to write two adjacent percent characters (%%). However, RSU cannot use environment variable names that contain percent characters. You can use the double-percent feature only outside environment variable names.

Environment variable names may contain spaces (example: %DEPT NAME%). It is not necessary to enclose them in quotes because of these spaces—RSU already recognizes the percent signs and processes them before spaces are considered. But superfluous quotes outside the variable name don't do any harm either.

Hint: Novell Netware 3.11 can place essential system information entries into the environment with the SET <variable>=<text> syntax of its login scripts. Example of a command in a Netware login script:

```
SET USER="%USER_ID"
```

DOS Commands

Any command found in an RSU program file that is not a valid RSU command is deemed to be a DOS command. A secondary command processor (COMMAND.COM) is loaded and the command forwarded to it and executed.

There is no error checking and no logging with DOS commands, so be careful to test and use them properly.

Echo

This command simply echoes some text on the screen. It works like the DOS command with the same name. It was incorporated only to increase speed and to avoid unnecessary empty lines on screen.

Syntax:

```
Echo <text>
```

Example:

```
Echo RSU is now installing new video drivers...
```

Goto

This command continues execution of the RSU program file at the point after the label. The label can be anywhere else in the program file but has to be in its own line, i.e. no other command can follow in the line that contains the label.

A label must be at least two characters long, not counting the colon. The reason for this is that c: means change to drive C:, rather than a label C. Labels may contain letters, digits and the underscore _. They may not contain spaces, umlauts or any other special characters.

Syntax:

```
Goto <label>
...
<label>:
```

Example:

```
Goto Finish
...
Finish:
```

If

This command executes the other commands embedded between If and End If only if the condition after the If is met.

In addition, an Else clause can be inserted. The commands after the Else and before the final End If are only executed if the condition after the If is not met.

The comparison operators < <= = >= > <> determine whether the character string to the left is less than, less or equal, equal, greater or equal, greater than, unequal to the string on the right of the operator, ignoring upper or lower case. Thus a = A would count as true.

The operator "Exist" followed by a filename determines whether the following file exists, similar to the equivalent DOS batch command.

The operator "Equal" between two filenames determines whether the two files are exactly equal in size, date and time.

A special form is the BIOS search. This allows to search a certain range of memory addresses within the first megabyte for a word. The syntax of the If clause is:

```
If Bios(<hexadr>--<hexadr>) = <text> Then
```

Example:

```
If Bios(F000-F100) = AMI Then
    Echo This computer has an AMI Bios.
End If
```

The BIOS search allows only the equal operator = and cannot be written with the search word on the left side of the equal sign. The search is case insensitive. Note also that you have to use double quotes if you want to search for more than one word, because the present syntax uses the space as a delimiter if not enclosed in double quotes. You may write:

```
If Bios(C000-C080) = "ati graphics ultra" Then
    Echo ATi Graphics Ultra found.
End If
```

Note that after the If and between any other elements on the line one or more spaces are needed. Thus it would be wrong to write:

```
If %USER%=DANIEL Then
```

The correct form is:

```
If %USER% = DANIEL Then
```

If any of the variables contains spaces, it must be enclosed in double quotes. Apart from that, an If command with a comparison operator must have exactly 5 words in the line.

Syntax (4 different forms):

```
If <condition> Then
    <command>
    ...
End If

If <condition> Then
    <command>
    ...
Else
    <command>
    ...
End If

If Not <condition> Then
    <command>
    ...
End If

If Not <condition> Then
    <command>
    ...
Else
    <command>
    ...
End If

<condition> = { <text 1> <comparison operator> <text 2> | Exist <filename> |
               <filename 1> Equal <filename 2> } | Bios(<hexadr>-<hexadr>) = <text>

<comparison operator> = { < | <= | = | >= | > | <> }

<hexadr> = 0 .. FFFF
```

Examples:

```
If Not %USER% = SUPERVISOR Then
    Goto Finish
...
Finish:

If Not c:\rsu\version.txt Equal f:\rsu\version.txt Then
    copy f:\rsu\version.txt c:\rsu
    ...
Else
    echo You already have the latest version. No update necessary.
EndIf

If Not Exist c:\windows\win.ini Then
    echo Your disk is not properly installed. Please follow
    echo the instructions to perform the base installation,
    echo then log on again.
    Goto Get_Out
End If

If Bios(F000-F200) = Dell Then
    c:\dell\keyb.com gr,,c:\dos\keyboard.sys
Else
    c:\dos\keyb.com gr,,c:\dos\keyboard.sys
End If
```

Hint: End If can be written with or without a space between End and If, i.e. either “End If” or “EndIf”. The preferred form is: “End If”.

Warning: Between each of two keywords, operators and operands one space is required.

Warning: Do not use the Goto command to jump into an If – End If structure. Do not use the Goto command to jump out of an If – End If structure many times, for example in a loop with more than a few repetitions.

IniAddLine

This command is used to add a line where several lines have the same variable name, like for example the device= lines in SYSTEM.INI. It appends one further line to the section. If the section does not exist it is newly created and appended to the .INI file first.

The only exception occurs if the exact line, variable name and text, exists in the file already. In this particular case the command has no effect. In other words, the command does not produce exact duplicates of whole lines like:

```
device=VSHARE.386
device=VSHARE.386
```

Syntax:

```
IniAddLine <ini file> [<section name>] <variable name>=<text>
```

Example:

```
IniAddLine C:\WIN31\SYSTEM.INI 386Enh device=VSHARE.386
```

IniChangeLine

This command changes the text after the equals sign (=) in a certain section and a certain line in an .INI file. If the section does not exist it is newly created and appended to the .INI file first. If the line does not exist it is newly created and appended at the end of the section. If several lines with the same variable name exist in the section then this command is probably not appropriate and should not be used since it would change only one of the lines.

Syntax:

```
IniChangeLine <ini file> [<section name>] <variable>=<text>
```

Example:

```
IniChangeLine C:\WIN31\WIN.INI [windows] load=NWPOPUP.EXE
```

Note that there is presently no command to change only part of a line. If something like this is desired one possible workaround is to use EDLIN in batch mode.

IniCopyLine

This command finds a certain line within a section in an .INI file and copies it into another .INI file. If a line with the same variable name to the left of the equals sign (=) already exists it is replaced with the new line. If several lines with the same variable name exist in the section then this command is probably not appropriate. It will work on the first occurrence of the variable. If the section does not exist in the target .INI file, it is newly created and appended to the .INI file first.

Syntax:

```
IniCopyLine <source ini file> <target ini file> [<section name>] <variable>
```

Examples:

```
IniCopyLine F:\RSU\WSIMAGE\WIN31\WIN.INI C:\WIN31\WIN.INI [windows] load
IniCopyLine F:\RSU\WSIMAGE\WIN31\WIN.INI C:\WIN31\WIN.INI [windows] load=
```

Both example lines do the same thing. Each one would search the file F:\RSU\WSIMAGE\WIN31\WIN.INI for the section [windows]. Within the section it would locate the line beginning with load= and copy it into the line with the same section and variable name in the file C:\WIN31\WIN.INI.

IniCopySection

This command works similar to the previous one but copies a whole section. If a section with the same name already exists in the target file, it is deleted and the new section copied and inserted in its place. IniCopySection also inserts an empty line before and after the section if there was none, to make the file easier to read and conform with the usual practice in Windows .INI files.

Syntax:

```
IniCopySection <source ini file> <target ini file> [<section name>]
```

Example:

```
IniCopySection F:\RSU\WSIMAGE\WIN31\WIN.INI C:\WIN31\WIN.INI [HPPCL5A,LPT1:]
```

This example would copy the whole section [HPPCL5A,LPT1:] from F:\RSU\WSIMAGE\WIN31\WIN.INI to C:\WIN31\WIN.INI. If there was a section with that name before it will be overwritten and all information lost entirely. If the previous section contained more or other lines than the new those old lines will be lost.

IniDeleteLine

This command deletes a line in an .INI file.

Syntax:

```
IniDeleteLine <ini file> [<section name>] <variable>  
IniDeleteLine <ini file> [<section name>] <variable>=  
IniDeleteLine <ini file> [<section name>] <variable>=<value>
```

Examples:

```
IniDeleteLine C:\WIN31\WIN.INI [mail] Polling  
IniDeleteLine C:\WIN31\WIN.INI [mail] Polling=  
IniDeleteLine C:\WIN31\SYSTEM.INI [386Enh] device=comm.drv
```

This example will search C:\WIN31\WIN.INI for the section [mail] and in this section for the line beginning with Polling=. This line will be deleted from WIN.INI. It will then search C:\WIN31\SYSTEM.INI, section [386Enh] and delete the line device=comm.drv. Note that, with this syntax, other device= lines are not affected. This makes it possible to change particular lines when the same variable occurs more than once, as usual with the device= lines in SYSTEM.INI.

IniDeleteSection

Syntax:

```
IniDeleteSection <ini file> [<section name>]
```

Example:

```
IniDeleteSection C:\WIN31\WIN.INI [Microsoft Word]
```

This example will search C:\WIN31\WIN.INI for the section [Microsoft Word]. The entire section will be deleted from WIN.INI.

SynchronizeDir

Makes the target directory equal to the source directory including all files, including files that have a read-only, hidden or system attribute.

SynchronizeDir handles all files regardless of their attributes. Attributes are copied with each file only if the Preserve subcommand is used. Without it the archive attribute is set and all other attributes are reset in the target file.

SynchronizeDir is a multi-line command and requires several subcommands to control its operation. Each subcommand has to be in one line. Empty lines and comment lines can also be used freely. The following subcommands can be used, and at least To and From and one other of them has to be used, otherwise SynchronizeDir will not do anything:

| | |
|-----------|---|
| From | Write the source directory after this subcommand. |
| To | Write the target directory after this subcommand. |
| Delete | Delete files from the target directory if they don't exist in the source directory. |
| Add | Add files to the target directory if they are not there already. |
| Overwrite | Overwrite files in the target directory if they also exist in the source directory but are different (have different size, date or time). This subcommand may not be used together with the Conflict subcommand. |
| Conflict | Conflict management. This subcommand may not be used together with the Overwrite subcommand. If a file name exists in both source and target directories but with different size, date or time, then this is considered a conflict and the following actions are taken: |

11. Both files are put into the target directory and the older one gets a new name like !SYNNnnn.xxx where nnnn is a number between 0001 and 9999 and xxx is the original extension of the file.
12. A line is appended to the file !SYN0000.TXT in the target directory containing the date, time and conflicting file information separated by semicolons (;), ready for import into a conflict database.

One possible purpose of this function is to allow users with portable PCs to copy their network user directories home with them and later reconcile their local user directories with those on the network if both can have changed.

Subdirectories Process subdirectories. All subdirectories of the directory to be synchronized are also processed.

Preserve Preserve attributes. With this subcommand the hidden, system and read-only attributes are copied from each source file to each destination file. Without this subcommand, these attributes are reset in the target file.

Warning: The Preserve subcommand affects only files that are copied for other reasons. It does not mean that existing equal files will now get their attributes copied if these are different. Keep in mind that the attributes are not used to determine whether two files are equal.

Report Write the report file name (complete with path, if desired) after this subcommand. Report all changes to a report text file. If the report text file already exists, the new records/lines are appended.

The report text file consists of one line for each reported file, followed by one carriage return and one line feed character. Each line contains the following 9 fields, separated by tab characters (character no. 9 in the ASCII/ANSI alphabet):

13. Transaction date in the format: YYYY-MM-DD Example: 1994-07-21
14. Transaction time in the 24 h format: HH:MM:SS Example: 23:59:30
15. Operation: ADD, DELETE, OV-ADD, OV-DEL, REN-ADD, REN-DEL. Since each line reports only one file, overwrite and rename operations (which work on two files) are reported in two lines, as if the file were first deleted, then added.

The first contains OV-DEL or REN-DEL and reports the disappearing file. The next contains OV-ADD or REN-ADD and reports the new file.

16. File or directory: FILE, DIR
17. Path and file name (if file). Examples: C:\SUBDIR\FILE.EXT, C:\SUBDIR
18. File/dir date in the format: YYYY-MM-DD Example: 1994-06-15
19. File/dir time in the 24 h format: HH:MM:SS Example: 13:35:50
20. File size in bytes (empty for directories). Example: 1735024
21. File attributes in the format: HSRA with minus signs in the place of attributes that are not set. Example for a file with read-only and archive attributes, but no hidden or system attributes: - - RA

ReportOnly Requires the presence of the Report subcommand (see above). No changes are actually made to directories or files. Only the report file is written (appended) as if the changes had been made. This can be used for testing or if the report file is evaluated by other means.

DirsLike Only one DirsLike statement can be included with any SynchronizeDir command. If it is present, only matching directories are processed. Write one directory name after this command. The name can contain the wildcard characters * and ? in the same way they are used in DOS commands.

FilesLike Only one FilesLike statement can be included with any SynchronizeDir command. If it is included, only matching files are processed. Write one file name after this command. The name can contain the wildcard characters * and ? in the same way they are used in DOS commands.

ExcludeDir Protect this directory and all its subdirectories from synchronization, i.e. do not read, compare, copy, change or delete it and do not synchronize it with anything. The directory can be given with full path and even the drive specified. If it is given without a full path, i.e. with neither a drive designator nor a backslash (\) at the beginning, then all directories with this name and all their subdirectories are excluded from synchronization if several occur within the directory tree. The ExcludeDir subcommand can occur several times with different subdirectories. It is useful only when the Subdirectories subcommand is also present.

ExcludeFile Protect this file from synchronization, i.e. do not read, compare, copy, change or delete it and do not synchronize it with anything. The file can be given with full path and even with the drive specified. If it is given without a full path, i.e. with neither a drive designator nor a backslash (\) at the beginning, then all files with this name are excluded from synchronization if several occur within the directory, or within the whole directory tree. The file name can contain the wildcard characters * and ? which have the same function as in DOS. The ExcludeFile subcommand can occur several times with different files.

End SynchronizeDir This ends the complete SynchronizeDir command. Exactly one space has to be between “End” and “SynchronizeDir”. This subcommand is always required after all others.

Warning: SynchronizeDir will overwrite or erase files with any attributes in the target directory and, with the Subdirectories subcommand, any of its subdirectories, even if they are read-only, hidden or system files.

Syntax:

```
SynchronizeDir
  From <source directory>
  To <target directory>
  [Delete]
  [Add]
  [{ Overwrite | Conflict }]
  [Subdirectories]
  [Preserve]
  [Report <report text file path and name>]
  [ReportOnly]
  [DirsLike <directory name with or without path and * or ? characters>]
  [FilesLike <file name with or without path and * or ? characters>]
  [ExcludeDir <directory>]
  ...
  [ExcludeFile <file name with or without path and * or ? characters>]
  ...
End SynchronizeDir
```

Examples:

```
SynchronizeDir
  From F:\RSU\WSIMAGE\U
  To C:\U
  Delete
  Overwrite
  Add
  Subdirectories
  FilesLike *.BA*
  ExcludeDir F:\RSU\WSIMAGE\U\ADMIN
  ExcludeFile *.BAK
  Report C:\SETUP\SYNCREP.TXT
  Preserve
End SynchronizeDir
SynchronizeDir
  From F:\RSU\WSIMAGE\DOS
  To C:\DOS
  Overwrite
  Add
End SynchronizeDir
```

The first example would make the directory C:\U and all of its subdirectories exactly equal to the directory F:\RSU\WSIMAGE\U and all of its subdirectories, as far as *.BA* files are concerned, except for the directory F:\RSU\WSIMAGE\U\ADMIN, which is entirely skipped. If a target directory C:\U\ADMIN exists, it is not touched either. All files with the extension .BAK are skipped. If files with the extension .BAK exist in the target directory, they also remain untouched. If files are copied, their file attributes are copied with them. A report text file C:\SETUP\SYNCREP.TXT is written, containing a list of all file changes.

The second would overwrite any files in C:\DOS that are different from their namesakes in F:\RSU\WSIMAGE\DOS. It would also add files that are missing in C:\DOS, but it would not delete or otherwise touch any additional files the user may have added to his DOS directory. It would also not touch any subdirectories of C:\DOS.

SynchronizeDir (Old Syntax)

This one-line version of the SynchronizeDir command is retained only for compatibility purposes and will be dropped from a future version of RSU. Please don't use it any more.

Note that the newer exclude subcommands and the ReportOnly subcommand are not available in this old syntax.

Makes the target directory equal to the source directory including all files, including files that have a read-only, hidden or system attribute.

SynchronizeDir handles all files regardless of their attributes. Attributes are copied with each file only if the /P switch is used. Without that switch the archive attribute is set and all other attributes are reset in the target file.

The SynchronizeDir command requires switches to control its operation. The following switches can be used, and at least one of them has to be used, otherwise SyncDir will not do anything:

- /D Delete files from the target directory if they don't exist in the source directory.
- /A Add files to the target directory if they are not there already.
- /O Overwrite files in the target directory if they also exist in the source directory but are different (have different size, date or time). This switch may not be used together with the /C switch.
- /C Conflict management. This switch may not be used together with the /O switch. If a file name exists in both source and target directories but with different size, date or time, then this is considered a conflict and the following actions are taken:

- 22. Both files are put into the target directory and the older one gets a new name like !SYNNnnn.xxx where nnnn is a number between 0001 and 9999 and xxx is the original extension of the file.
- 23. A line is appended to the file !SYN0000.TXT in the target directory containing the date, time and conflicting file information separated by semicolons (;), ready for import into a conflict database.

One possible purpose of this function is to allow users with portable PCs to copy their network user directories home with them and later reconcile their local user directories with those on the network if both can have changed.

- /S Process subdirectories. All subdirectories of the directory to be synchronized are also processed.
- /R Report all changes to a report text file. The filename (including path if desired) has to follow the /R switch either with or without an intervening space. If the report text file already exists, the new records/lines are appended.

The report text file consists of one line for each reported file, followed by one carriage return and one line feed character. Each line contains the following 9 fields, separated by tab characters (character no. 9 in the ASCII/ANSI alphabet):

- 24. Transaction date in the format: YYYY-MM-DD Example: 1994-07-21
- 25. Transaction time in the 24 h format: HH:MM:SS Example: 23:59:30
- 26. Operation: ADD, DELETE, OV-ADD, OV-DEL, REN-ADD, REN-DEL. Since each line reports only one file, overwrite and rename operations (which work on two files) are reported in two lines, as if the file were first deleted, then added. The first contains OV-DEL or REN-DEL and reports the disappearing file. The next contains OV-ADD or REN-ADD and reports the new file.
- 27. File or directory: FILE, DIR
- 28. Path and file name (if file). Examples: C:\SUBDIR\FILE.EXT, C:\SUBDIR
- 29. File/dir date in the format: YYYY-MM-DD Example: 1994-06-15
- 30. File/dir time in the 24 h format: HH:MM:SS Example: 13:35:50

31. File size in bytes (empty for directories). Example: 1735024

32. File attributes in the format: HSRA with minus signs in the place of attributes that are not set. Example for a file with read-only and archive attributes, but no hidden or system attributes: --RA

/P Preserve attributes. With this switch the hidden, system and read-only attributes are copied from each source file to each destination file. Without this switch, these attributes are reset in the target file.

Warning: The /P parameter affects only files that are copied for other reasons. It does not mean that existing equal files will now get their attributes copied if these are different. Keep in mind that the attributes are not used to determine whether two files are equal.

Warning: SynchronizeDir will overwrite or erase files with any attributes in the target directory and, with the /S switch, any of its subdirectories, even if they are read-only, hidden or system files.

Syntax:

```
SynchronizeDir <source directory> <target directory> [/D] [< /O | /C >] [/A] [/S]
[/R <report text file path and name>] [/P]
```

Examples:

```
SynchronizeDir F:\RSU\WSIMAGE\U C:\U /D /O /A /S /R C:\SETUP\SYNCREP.TXT /P
SynchronizeDir F:\RSU\WSIMAGE\DOS C:\DOS /O /A
```

The first line would make the directory C:\U and all of its subdirectories exactly equal to the directory F:\RSU\WSIMAGE\U and all of its subdirectories. If files are copied, their file attributes are copied with them. A report text file C:\SETUP\SYNCREP.TXT containing all file changes is written.

The second would overwrite any files in C:\DOS that are different from their namesakes in F:\RSU\WSIMAGE\DOS. It would also add files that are missing in C:\DOS, but it would not delete or otherwise touch any additional files the user may have added to his DOS directory. It would also not touch any subdirectories of C:\DOS.

Brief Command Overview

In the following text each group of syntax lines is followed by one or more example lines which are indented.

```
RSU <program file name>
RSU.EXE <program file name>
RSU <program file name> /debug
RSU.EXE <program file name> /debug
    f:
    cd \rsu
    rsu rsu.prg

Echo <text>
    Echo RSU is now installing new video drivers...

Goto <label>
    Goto Finish

<label>:
    Finish:
```

```

If [Not] <condition> Then
    <command>
    ...
[Else
    <command>
    ...]
End If

If [Not] <condition> Then
    <command>
    ...
[Else
    <command>
    ...]
EndIf

<condition> = { <text 1> <comparison operator> <text 2> | Exist <filename> |
    <filename 1> Equal <filename 2> } | Bios(<hexadr>-<hexadr>) = <text>

<comparison operator> = { < | <= | = | >= | > | <> }

<hexadr> = 0 .. FFFF

    If Bios(F000-F100) = AMI Then
        Echo This computer has an AMI Bios.
    End If
    If Not %USER% = SUPERVISOR Then
        Goto Finish
    End If
    ...
Finish:

```

(The example above also shows how to insert an environment variable.)

```

IniAddLine <ini file> [<section name>] <variable name>=<text>
IniAddLine C:\WIN31\SYSTEM.INI 386Enh device=VSHARE.386
IniAddLine C:\WIN31\SYSTEM.INI 386Enh device=NETWARE.386

IniChangeLine <ini file> [<section name>] <variable>=<text>
IniChangeLine C:\WIN31\WIN.INI [windows] load=NWPOPUP.EXE
IniChangeLine C:\WIN31\WIN.INI [mail] mailbox=%USER%

```

(The second example above also shows how to insert an environment variable.)

```

IniCopyLine <source ini file> <target ini file> [<section name>] <variable>
IniCopyLine F:\RSU\WSIMAGE\WIN31\WIN.INI C:\WIN31\WIN.INI [windows] load
IniCopyLine F:\RSU\WSIMAGE\WIN31\WIN.INI C:\WIN31\WIN.INI [windows] load=

IniCopySection <source ini file> <target ini file> [<section name>]
IniCopySection F:\RSU\WSIMAGE\WIN31\WIN.INI C:\WIN31\WIN.INI [HPPCL5A,LPT1:]

IniDeleteLine <ini file> [<section name>] <variable>
IniDeleteLine <ini file> [<section name>] <variable>=
IniDeleteLine <ini file> [<section name>] <variable>=<value>
IniDeleteLine C:\WIN31\WIN.INI [mail] Polling
IniDeleteLine C:\WIN31\WIN.INI [mail] Polling=
IniDeleteLine C:\WIN31\SYSTEM.INI [386Enh] device=comm.drv

IniDeleteSection <ini file> [<section name>]
IniDeleteSection C:\WIN31\WIN.INI [Microsoft Word]

```



```

SynchronizeDir
  From <source directory>
  To <target directory>
  [Delete]
  [Add]
  [{ Overwrite | Conflict }]
  [Subdirectories]
  [Preserve]
  [Report <report text file path and name>]
  [ReportOnly]
  [DirsLike <directory name with or without path and * or ? characters>]
  [FilesLike <file name with or without path and * or ? characters>]
  [ExcludeDir <directory>]
  ...
  [ExcludeFile <file name with or without path and * or ? characters>]
  ...
End SynchronizeDir
  SynchronizeDir
    From F:\RSU\WSIMAGE\U
    To C:\U
    Delete
    Overwrite
    Add
    Subdirectories
    FilesLike *.BA*
    ExcludeDir F:\RSU\WSIMAGE\U\ADMIN
    ExcludeFile *.BAK
    Report C:\SETUP\SYNCREP.TXT
    Preserve
  End SynchronizeDir
  SynchronizeDir
    From F:\RSU\WSIMAGE\DOS
    To C:\DOS
    Overwrite
    Add
  End SynchronizeDir

```

Other features:

%<environment variable>% is replaced by the contents of the environment variable.

%% is replaced by %

"<any text containing spaces>" is taken as one parameter.

"" is replaced by "