

MKMSGF

Brief

Build message file from message source file

Syntax

```
drive path MKMSGF drive path
```

MKMSGF infile[.ext] outfile[.ext] [/?] [/V] [/D <DBCS range or country>] [/P <code page>] [/L <language id,sub id>]

Arguments

- infile - the input file that contains message definitions. The input-file name can be any valid OS/2 file name, optionally preceded by a drive letter and a path.
- outfile field specifies the output file created by MKMSGF. The output-file name can be any valid OS/2 file name, optionally preceded by a drive letter and a path.

A valid DBCS range is: n10,n11,n20,n21,...,nn0,nn1

A single number is taken as a DBCS country code.

The valid OS/2 language/sublanguage ID values are:

| Language ID: | | | | |
|--------------|--------|-----|---------------------|-------------------|
| Code | Family | Sub | Language | Principal country |
| ---- | ----- | --- | ----- | ----- |
| ARA | 1 | 2 | Arabic | Arab Countries |
| BGR | 2 | 1 | Bulgarian | Bulgaria |
| CAT | 3 | 1 | Catalan | Spain |
| CHT | 4 | 1 | Traditional Chinese | R.O.C. |
| CHS | 4 | 2 | Simplified Chinese | P.R.C. |
| CSY | 5 | 1 | Czech | Czechoslovakia |
| DAN | 6 | 1 | Danish | Denmark |
| DEU | 7 | 1 | German | Germany |
| DES | 7 | 2 | Swiss German | Switzerland |
| EEL | 8 | 1 | Greek | Greece |
| ENU | 9 | 1 | US English | United States |
| ENG | 9 | 2 | UK English | United Kingdom |
| ESP | 10 | 1 | Castilian Spanish | Spain |
| ESM | 10 | 2 | Mexican Spanish | Mexico |
| FIN | 11 | 1 | Finnish | Finland |
| FRA | 12 | 1 | French | France |

| | | | | |
|-----|----|---|---------------------|----------------|
| FRB | 12 | 2 | Belgian French | Belgium |
| FRC | 12 | 3 | Canadian French | Canada |
| FRS | 12 | 4 | Swiss French | Switzerland |
| HEB | 13 | 1 | Hebrew | Israel |
| HUN | 14 | 1 | Hungarian | Hungary |
| ISL | 15 | 1 | Icelandic | Iceland |
| ITA | 16 | 1 | Italian | Italy |
| ITS | 16 | 2 | Swiss Italian | Switzerland |
| JPN | 17 | 1 | Japanese | Japan |
| KOR | 18 | 1 | Korean | Korea |
| NLD | 19 | 1 | Dutch | Netherlands |
| NLB | 19 | 2 | Belgian Dutch | Belgium |
| NOR | 20 | 1 | Norwegian - Bokmal | Norway |
| NON | 20 | 2 | Norwegian - Nynorsk | Norway |
| PLK | 21 | 1 | Polish | Poland |
| PTB | 22 | 1 | Brazilian Portugues | Brazil |
| PTG | 22 | 2 | Portuguese | Portugal |
| RMS | 23 | 1 | Rhaeto-Romanic | Switzerland |
| ROM | 24 | 1 | Romanian | Romania |
| RUS | 25 | 1 | Russian | U.S.S.R. |
| SHL | 26 | 1 | Croato-Serbian (Lat | Yugoslavia |
| SHC | 26 | 2 | Serbo-Croatian (Cyr | Yugoslavia |
| SKY | 27 | 1 | Slovakian | Czechoslovakia |
| SQI | 28 | 1 | Albanian | Albania |
| SVE | 29 | 1 | Swedish | Sweden |
| THA | 30 | 1 | Thai | Thailand |
| TRK | 31 | 1 | Turkish | Turkey |
| URD | 32 | 1 | Urdu | Pakistan |
| BAH | 33 | 1 | Bahasa | Indonesia |
| SLO | 34 | 1 | Slovene | Slovenia |

For a complete list of code pages and country codes, see the code page table under COUNTRYCODE in the online book Control Program Programming Guide and Reference.

Input Message File Format

The input message file is a standard ASCII file that contains three types of lines:

- Comment lines
- Component identifier line
- Component message lines

Comment Lines

Comment lines are allowed anywhere in the input message file, except between the component identifier and the first message. Comment lines must begin with a semicolon (;) in the first column.

In the Input Message File Example, the comment lines are

```
; This is a sample of an input
; message file for component DOS
; starting with three comment lines.
Component Identifier Line
```

The component-identifier line contains a three-character name identifier that precedes all MKMSGF message numbers.

In the example, the component identifier is DOS. Component-Message Lines

Each component-message line consists of a message header and an ASCII text message.

The message header is comprised of the following parts:

```
A three-character component identifier
A four-digit message number
A single character specifying message type (E, H, I, P, W, ?)
A colon (:)
Followed by a blank space.
```

The following message types are used:

| Type | Meaning |
|------|------------------------------------|
| E | Error |
| H | Help |
| I | Information |
| P | Prompt |
| W | Warning |
| ? | no message assigned to this number |

Message numbers can start at any number, but messages must be numbered sequentially. If you do not use a message number, you must insert an empty entry in its place in the text file. An empty entry consists of the message number, with ? as the message type, and no text.

The character % has a special meaning when used within the text of a message:

%0 is placed at the end of a prompt (type P) to prevent DosGetMessage from executing a carriage return and line feed. This allows the user to be prompted for input on the same line as the message text.

Mike Note: The %0 can be used with any message type!!!! So here is how this works: The message file is scanned, and each is saved. However, if you place a %0 as the last character the <LF> <CR> is dropped and it does not matter the type of message (E, H, I, P, W, or ?). This is how the IBM MKMSGF worked and my clone works the same way.

%1 - %9 are used to identify variable string insertion within the text of a message. These variables correspond to the ltable and lvCount parameters in the DosGetMessage call.

Component-Message Example

For example, DOS0100E: is DOS error message 100. For additional examples, see the Input Message File Example.

Output File

The output file contains the indexed message file that DosGetMessage will use. The output-file name can be any valid OS/2 file name, optionally preceded by a drive letter and a path. The output file cannot have the same name as the input file.

To differentiate between the two files, the following convention is recommended, using the same file name.

```
The infile file should have a .TXT extension.  
The outfile file should have a .MSG extension.
```

Help-message file names begin with the component identifier, followed by H.MSG. For example, the help file associated with the component identifier DOS would be DOSH.MSG.

Options

Text-based messages in different code pages can be created using MKMSGF to display errors, help information, prompt, or provide general information to the application user.

MKMSGF uses the following parameter formats to build message files:

```
MKMSGF infile outfile /Pcodepage  
MKMSGF infile outfile /Ddbcsrange or country id  
MKMSGF infile outfile /LlangID,VerId  
MKMSGF infile outfile /V  
MKMSGF infile outfile /?  
MKMSGF @controlfile
```

```
Infile is the ASCII-text source file.
```

Example:

```
MSG  
MSG0001I: (mm%4dd%4yy) %2%4%1%4%3  
MSG0002I: (dd%4mm%4yy) %1%4%2%4%3  
MSG0003I: Current date is: %0
```

%0 is a special argument that displays a prompt for user input. %1 - %9 are the arguments the user can use to insert text in a message.

```
Outfile is the binary output message file.  
@controlfile is the message definition file.
```

Options Summary

| Type | Meaning |
|------|--|
| /P | Code-page ID for the input message file. |
| /D | DbscRange or country ID for the input message file. |
| /L | Language family ID (one word) and language version ID (one word). |
| /V | Verbose display of message file control variables as the message file is |

```
being created.  
/? Help display of command syntax for MKMSGF.
```

Note: Any combination of /P, /D, /L, and /V switches can be used for either the command line or @controlfile execution method.

The / switch prefix and the - prefix can be used interchangeably when defining switches to MKMSGF.

/Verbose Option Output Example

Following is a sample of MKMSGF output, using the Verbose option (/V). This output was produced using the following command:

```
mkmsgf myapp.txt myapp.msg /v
```

```
strIn      = myapp.txt  
strOut     = myapp.msg  
StrIncDir  = (null)  
CodePages  = 437  
Language family id = 0 and sub id = 0  
Language family id and sub id = unspecified  
flags     = none  
CP_type   = SBCS  
"myapp.txt": length = 382 bytes.  
29 messages scanned. Writing output file...  
Size of table entry: word
```

/P Option

The Code-page option (/P) specifies the code-page ID for that input message file.

For a complete list of code pages, see the code page table under COUNTRYCODE in the online book Control Program Programming Reference.

Up to 16 /P combinations can be saved with the message file.

/P cannot be used to identify DBCS data.

/D Option

The DBCS option (/D) specifies the DBCS Range or country ID for that input message file.

A valid DBCS country ID will cause the initialization of valid DBCS ranges to be set up for this file.

See DBCS Code Pages and Country Codes for valid DBCS country codes. /L Option

The Language option (/L) specifies the language family ID (one word) and language version ID (one word).

Valid combination of language family and language version will be set for this file.

A valid language family with invalid or undefined language version id will cause a default value of 1 to be set for language version. /A /C /I Options

I saw these and documented them as existing years ago, but never really looked at them in depth. I made some notes on a separate page: The A, C, and I Options.

/E

See the The Extended Structure page for more information. This tacks on what I call a fake extended header at the end of the file and updates the header offset. Control Files

The control file (@controlfile) is used to create multiple-code-page message files. The at sign (@) is not part of the file name, but rather, a delimiter required before a control-file name.

The control file has the following format:

Example:

```
root.in root. Out /Pcodepage /Ddbcsrang/ctryid /LlangID,VerId
sub.001 sub1.out /Pcodepage /Ddbcsrang/ctryid /LlangID,VerId
      .
      .
sub.00n subn.out /Pcodepage /Ddbcsrang/ctryid /LlangID,VerId
```

The help option (/?) is invalid due to the purpose of the definition file.

Note: Any combination of /P /D /L and /V switches can be used for either the command line or msg_definition_file execution method. Input Message File Example

Following is an example of an input message file:

```
; This is a sample of an input
; message file for component MAB
; starting with three comment lines.
MAB
MAB0100E: File not found
MAB0101?:
MAB0102H: Usage: del [drive:][path] filename
MAB0103?:
MAB0104I: %1 files copied
MAB0105W: Warning! All data will be destroyed!
MAB0106?:
MAB0107?:
MAB0108P: Do you wish to apply these patches (Y or N)? %0
MAB0109E: Divide overflow
```

Notes

Text based on <https://github.com/MikeyG/mkmsgf/wiki/MKMSGF-Usage>

From:
<http://www.osfree.org/doku/> - **osFree wiki**

Permanent link:
<http://www.osfree.org/doku/doku.php?id=en:docs:tk:tools:mkmsgf&rev=1705653234>

Last update: **2024/01/19 08:33**

