



Sophisticated 2D Structure
Visualization Tool Generating Various
Graphics File Formats

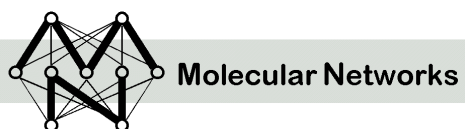
User Manual

Version 1.0

for program version 1.0 (or higher)

November 2003

Molecular Networks GmbH – Computerchemie
Nägelsbachstraße 25, 91052 Erlangen, Germany



Molecular Networks GmbH

Computerchemie

Nägelsbachstr. 25

91052 Erlangen

Germany

Phone: +49-(0)9131-815668

Fax: +49-(0)9131-815669

Email: info@mol-net.de

WWW: www.mol-net.de

This document is copyright © 2002 by Molecular Networks GmbH Computerchemie. All rights reserved. Except as permitted under the terms of the Software Licensing Agreement of Molecular Networks GmbH Computerchemie, no part of this publication may be reproduced or distributed in any form or by any means or stored in a database retrieval system without the prior written permission of Molecular Networks GmbH Computerchemie.

The software described in this document was originally designed and implemented for the CACTVS system by W. D. Ihlenfeldt. It is not part of the standard CACTVS toolkit distribution; it is furnished under a license and may be used and copied only in accordance with the terms of such license. For licensing this software please contact Molecular Networks GmbH, Nägelsbachstr. 25, 91052 Erlangen, Germany.

Product names and company names may be trademarks or registered trademarks of their respective owners, in the Federal Republic of Germany and other countries. All rights reserved.

Table of Contents

1.	General Information about MN.IMAGE.....	5
2.	Use cases	5
2.1.	High-quality 2D visualization of chemical structures	5
2.2.	Embedding structures into text documents	5
2.3.	Embedding structures into intranet or internet documents	6
3.	Installation	7
3.1.	Requirements.....	7
3.2.	Installation steps for UNIX operating systems (IRIX, Solaris, Linux)	7
3.3.	Installation steps for Windows operating systems (NT4/2000/XP).....	7
4.	Uninstallation.....	7
4.1.	Uninstallation steps for UNIX operating systems (IRIX, Solaris, Linux).....	7
4.2.	Uninstallation steps for Windows operating systems (NT4/2000/XP)	7
5.	Problems and Help!.....	8
6.	Release Notes	9
6.1.	Version 1.0	9
7.	Getting Started.....	10
7.1.	UNIX operating systems.....	10
7.2.	Windows operating systems.....	10
8.	Program Use	11
8.1.	Synopsis	11
8.2.	General program features.....	12
8.3.	Supported file formats for input files.....	13
8.4.	Program features for file handling.....	13
-format gif/png/emf/wmf/pwmf/eps/bmp24/bmp8/bmp8c/bmp4c/bmp1.....	13	
-name record/name	14	
-directory <directory name>	14	
-cleardirectory	15	
-count <n>	15	
-offset <n>.....	15	
-feedback.....	15	
-antialiasing 0/1	16	
-interlace 0/1.....	16	
-copyright <text>.....	16	
-embed none/smiles	17	
-metadata 0/1	17	
8.5.	Program features for setting the layout of the image.....	18
-height <pixels>.....	18	
-width <pixels>	18	
-background <colorname>/transparent.....	19	
-border <pixels>	19	
-crop <nborderpixels>	20	

-header <text>	20
-headerproperty <propertyname>	21
-headercolor <colorname>	22
-footer <text>	22
-footerproperty <propertyname>	23
-footercolor <colorname>	24
-logfile <filename>	24
-logoscale <factor>	24
8.6. Program features for setting the layout of molecules	26
-asymbol symbol/xsymbol/label/index/box/compact	26
-hsymbol none/special/all	26
-csymbol none/special/all	27
-atomcolor <colorname>/type	28
-hcolor <colorname>	28
-bondcolor <colorname>/split	29
-symbolfontsize <pts>	29
-annotationfontsize <pts>	30
-font fontfile	31
-bondscalescale <n>	31
-bead 0/1	32
-showcharge <0/1>	32
-showisotope <0/1>	33
-showradical <0/1>	33
-showstereo <0/1>	34
-showstereoh <0/1>	34
-wedges 0/1	35
-dashes 0/1	35
8.7. Program features of general usage	36
-version	36
9. Extended Features only available for the UNIX operating systems	36
10. Application examples	37
10.1. Table of 20 natural amino acids	37
11. Frequently Asked Questions (FAQ)	38
12. Error messages	38
13. Known problems and limitations	38
14. Technical Support	39
The MN.IMAGE Web Site	39
Reporting Problems	39
Updates	39
Contact information	39
15. Report Form	40
16. Index	41

1. General Information about MN.IMAGE

MN.IMAGE is a powerful batch-mode processor for chemical structure files. It converts a variety of chemical file formats into 2D pixel images (GIF, PNG, BMP, etc.) or vector drawings (WMF, EMF, etc.).

The program MN.IMAGE

- supports several options for setting the layout (background color, color of atom symbols, etc.)
- processes datasets with 99.9% conversion rate
- handles datasets of hundreds of thousands of chemical structures
- supports over 10 graphics file formats (GIF, PNG, EMF, WMF, BMP, EPS, ...)

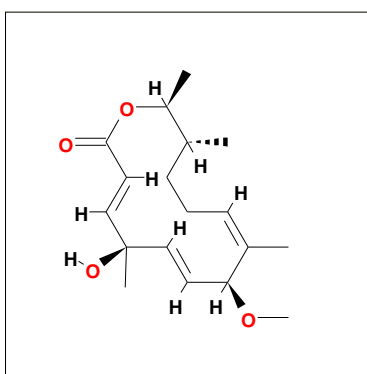
2. Use cases

Nowadays, there are quite a lot of programs available generating chemical structures in electronic form. Apart from storing this information, chemists also need programs for retrieving and visualizing this valuable information. One way of visualizing chemical structures is the generation of 2D structure images. As the program supports both pixel image formats and vector drawing formats these generated pictures can be embedded in various program applications, such as word processing programs or even in internet or intranet solutions.

As MN.IMAGE is a very reliable structure depiction program it could be used whenever high-quality 2D pixel images or vector drawings are essential.

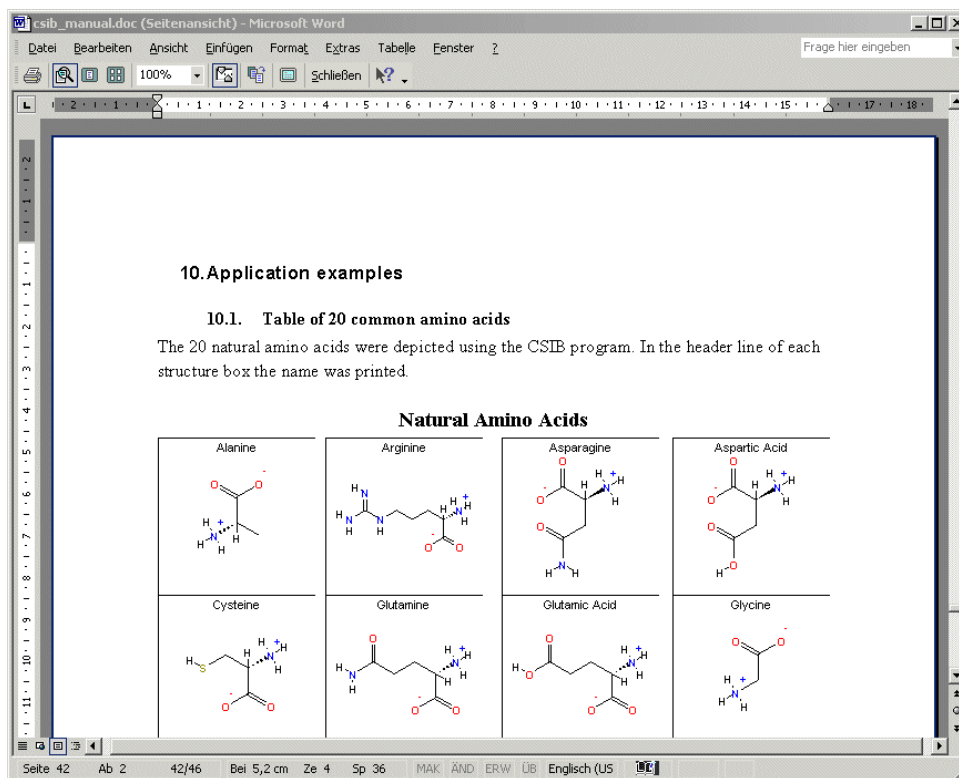
2.1. High-quality 2D visualization of chemical structures

MN.IMAGE is a sophisticated 2D structure visualization tool generating 2D structure drawings in various graphics file formats. In comparison with other similar programs, MN.IMAGE provides an enhanced set of features and superior layout quality.



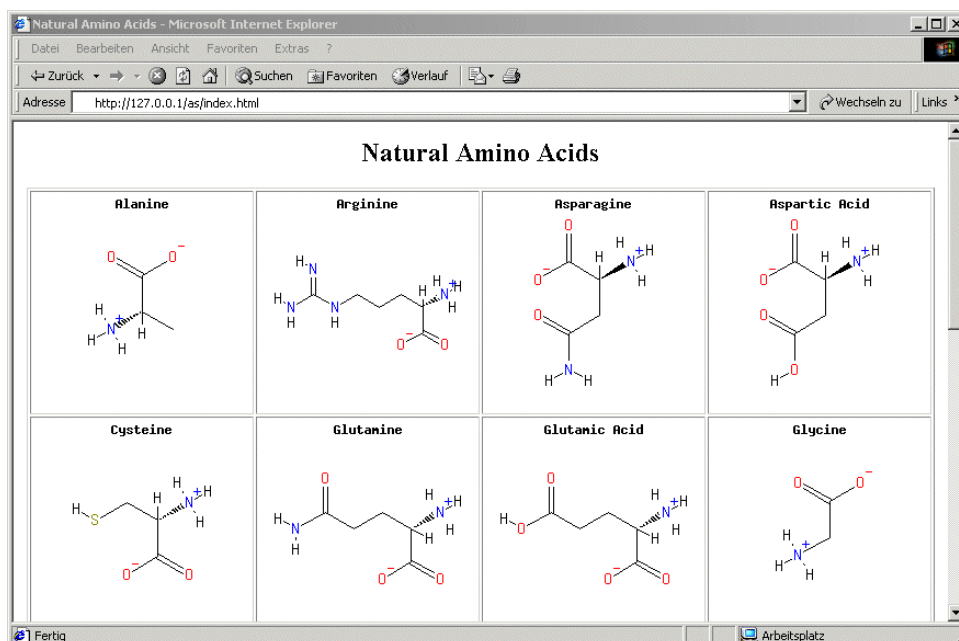
2.2. Embedding structures into text documents

Generated structures can easily be embedded into text documents. As MN.IMAGE is also able to generate vector graphics formats the size of the image can be adapted without any loss in quality.



2.3. Embedding structures into intranet or internet documents

Generated pixel images of chemical structures, such as GIF and PNG, can be embedded into websites shown in the following picture.



3. Installation

3.1. Requirements

MN.IMAGE is available for common UNIX platforms (x86 Linux, Sun Solaris, SGI IRIX). It is also available for Microsoft Windows NT4/2000/XP.

The program is running in a batch mode.

3.2. Installation steps for UNIX operating systems (IRIX, Solaris, Linux)

- 1.) Create a subdirectory, e.g., `mn_image`
(for system administrators when installing software locally, e.g. `/usr/local/bin/mn_image`).
- 2.) Copy the file `mn_image_<version>.<os>.gz` to the subdirectory `mn_image`
- 3.) Unpack the distribution by executing the `gunzip` command:
`gunzip mn_image_<version>.<os>.gz`
- 4.) Rename the file `mn_image_<version>.<os>` to `mn_image`.
Please note: `mn_image_<version>.<os>` is a binary file.
- 5.) Add the `mn_image` subdirectory name to the environment variable `PATH` in your `.login` or `.cshrc` files (`.profile` or `.bashrc`).

Launch MN.IMAGE with the command

```
mn_image -version or  
/usr/local/bin/mn_image/mn_image -version
```

3.3. Installation steps for Windows operating systems (NT4/2000/XP)

Although administrator privileges are not necessary, we recommend logging in as administrator. Double-click on the executable setup program and follow the instructions on the screen.

After successful installation there is no need to reboot your PC.

4. Uninstallation

4.1. Uninstallation steps for UNIX operating systems (IRIX, Solaris, Linux)

Log in as root and delete the file `mn_image` in your installation directory carefully (default path during installation was `/usr/local/bin/mn_image/`).

4.2. Uninstallation steps for Windows operating systems (NT4/2000/XP)

Log in as administrator, launch the uninstaller and follow the on-screen instructions.

5. Problems and Help!

If you have any difficulties with the installation of MN.IMAGE or if any problems occur while running MN.IMAGE, please send all your inquiries to the following address:

Molecular Networks GmbH Computerchemie
Nägelsbachstr. 25
91052 Erlangen
Germany,

or contact us by email
or by fax

support@mol-net.de
+49-(0)9131 - 81 56 69.

Please mention the program version of MN.IMAGE (`mn_image -version`), include your input file, and the output file on an MS/DOS diskette (3½") or send it to us by email. These files will help us to analyze the problem; if your system displays any error messages, please add them to your report.

You can also use the report form at the end of this manual.

6. Release Notes

6.1. Version 1.0

First release of MN.IMAGE

7. Getting Started

7.1. UNIX operating systems

The example file `six_examples.sdf` submitted with the distribution contains the structure information of six molecules in SDF format. Copy this example file into your working directory and type the following command:

```
mn_image -format wmf six_examples.sdf
```

MN.IMAGE now creates the output files named `six_examples_00001.wmf`, `six_examples_00002.wmf`, etc. within the new directory `six_examples`. Figure 1 shows two images of the output file stored as Windows Metafile which were embedded into this document.

If you have no permission writing to the directory in which the program was installed, set the **-directory** option for specifying another directory:

```
mn_image -format wmf -directory /tmp six_examples.sdf
```

7.2. Windows operating systems

The example file `six_examples.sdf` submitted with the distribution contains the structure information of six molecules in SDF format. Copy this example file into your working directory and open a DOS shell. Change the working directory to the directory where you installed `mn_image` by using the `cd` command then type the following command:

```
mn_image -format wmf six_examples.sdf
```

MN.IMAGE now creates the output files named `six_examples_00001.wmf`, `six_examples_00002.wmf`, etc. within the new directory `six_examples`. Figure 1 shows two images of the output file stored as Windows Metafile which were embedded into this document.

If you have no permission writing to the directory in which the program was installed, set the **-directory** option for specifying another directory:

```
mn_image -format wmf -directory C:\temp six_examples.sdf
```

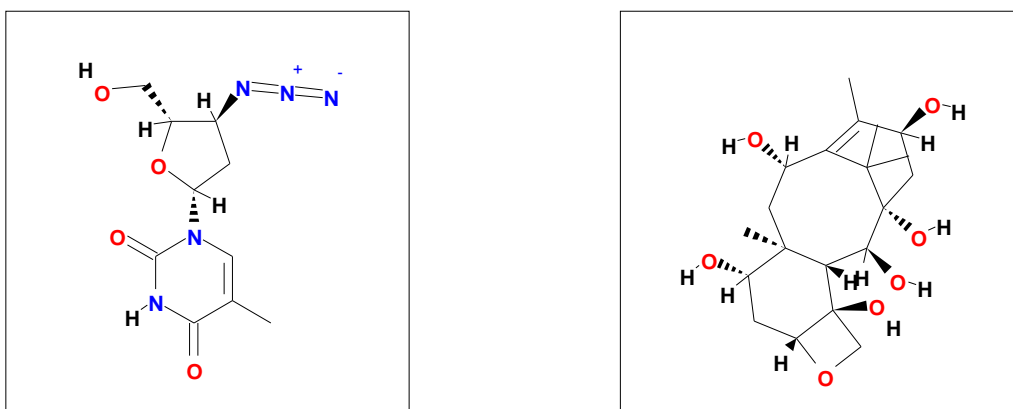


Figure 1: 2D image of the structure no. 1 and 6 of the input file.

8. Program Use

8.1. Synopsis

The general synopsis for using MN.IMAGE is:

```
mn_image [ -option(s) ] [ infile(s) ]
```

An overview of the various options is given in Table 1 and in more detail in the following chapter. Infile is the input file name. If no file names are given, the program reads from standard input. By default, the images of every file are put into a separate subdirectory. The name of the subdirectory is the file name without any suffix, in the current working directory, not where the file resides.

[-antialiasing 0/1]	[-format gif/png/emf/wmf/pwmf/eps/bmp24/bmp8/bmp8c/bmp4/bmp4c/bmp1]
[-annotationfontsize pts]	[-hcolor colorname]
[-asymbol symbol/xsymbol/label/index/compact]	[-header text]
[-atomcolor colorname/type]	[-headercolor colorname]
[-background colorname/transparent]	[-headerproperty propertyname]
[-bead 0/1]	[-height pixels]
[-bondcolor colorname/split]	[-hsymbol none/special/all]
[-bondscale n]	[-interlace 0/1]
[-border pixels]	[-logfile filename]
[-cleardirectory]	[-logoscale factor]
[-copyright text]	[-metadata 0/1]
[-count n]	[-name record/name]
[-crop nborderpixels]	[-offset records]
[-csymbol all/special/none]	[-showcharge 0/1]
[-dashes 0/1]	[-showisotope 0/1]
[-directory dirname]	[-showradical 0/1]
[-embed none/smiles]	[-showstereo 0/1]
[-feedback]	[-showstereoh 0/1]
[-font fontfile]	[-symbolfontsize points]
[-footer text]	[-version]
[-footercolor colorname]	[-wedges 0/1]
[-footerproperty propertyname]	[-width pixels]

Table 1: Overview of all options supported by MN.IMAGE.

8.2. General program features

MN.IMAGE will read the specified input files, perform selected operations on the input data, and produces either pixel images (GIF, PNG, BMP, etc.) or vector drawings (EPS, WMF, EMF, etc.) of the structures.

The file type of the input files is automatically recognized. It may change from file to file. If no input file is specified, or the file name „-“ is used, the program reads from standard input. Regular expressions used to specify a set of input files, like “mol*.sdf”, are also allowed. If, instead of a file name, the name of a directory is specified, all readable structure files of known formats in that directory are read as a single virtual data file. If you are running MN.IMAGE under a UNIX operating system, there are some more features for reading input files (see chapter 9 “Extended Features only available for the UNIX operating systems” for more details).

By default, the images of every file are put into a separate subdirectory. The name of the subdirectory is the file name without any suffix, in the current working directory, not where the file resides. If the file does not have a suffix, the suffix `_img` is added to the directory name. The target directory can be overridden by the `-directory` option. In this case, all images are put into that directory.

The names of the images are by default constructed from the root name of the respective input file (without directory part, and without suffix), followed by the record count, and the standard image file format suffix (`.gif`, `.png`, `.wmf`, etc.). Thus, an input file `/usr/local/test/test.sdf` will by default produce images named `test_00001.gif`, `test_00002.gif`, etc. in the subdirectory `test` of the current working directory. See the `-name` option for an alternative naming scheme.

Structures are automatically scaled to fit completely into images of arbitrary size. Structures which possess up to a configurable number of standard-length bonds in x-direction are scaled by a constant factor and centered in the image. Larger structures are dynamically shrunk. This algorithm generates pleasant plots of average molecules with plot sizes directly proportional to the structure size, while avoiding bad plots with incomplete, overlapping drawings. The length of a standard bond is intelligently determined from a plot coordinate analysis. The fonts for the displays of atom symbols are automatically selected according to the scaling factor. If the scaling factor becomes very small, the atom symbols are replaced by small, appropriately colored squares.

For pixel images, the program has two different choices of fonts to render all texts (atom symbols, charges, header and footer lines, etc.). First, it has a small collection of rather blocky built-in fonts. These are used by default. Alternatively, any TrueType font can be used. The standard Linux TrueType fonts Arial, Arialn (narrow), Arialb (bold) and Arialnb (narrow bold) are included in the standard distribution. More fonts can be added by copying them into the fonts distribution directory. For vector images (EPS, EMF, WMF) no fonts need to be actually present. See the `-font` option for selecting a special TrueType font.

8.3. Supported file formats for input files

The program will automatically detect the file format of the input files. Two standard exchange formats are supported. Thus, there is no need for a parameter specifying the input format. A set of input files does not need to have a common format.

The supported file formats are listed in the table below. Please note, that the image generation of reaction files is currently not supported.

Full Format Name	Default Input-Extension	Read	Comment
MDL Molfile	mol	Yes	
MDL SDF	sdf	Yes	

Table 2: Overview of the supported input file formats

8.4. Program features for file handling

-format gif/png/emf/wmf/pwmf/eps/bmp24/bmp8/bmp8c/bmp4c/bmp1

The parameter **-format** sets the output format.

The **GIF** (Graphics Interchange Format) encoding uses a GIF-compatible, but not patented algorithm for the image data compression. The disadvantage of this encoding is that it is notably less efficient than the original protected procedure, so the GIF images tend to be large for their content. They can, however, be re-encoded by a licensed image converter. Both **GIF** and **PNG** (Portable Network Graphics) can store additional, invisible data, such as a copyright message (please see the parameter **-copyright**), encoded structure information (parameter **-embed**) and image metadata (parameter **-metadata**). However, care must be taken when post-processing these images: many image manipulation programs do not support the handling of auxiliary data, which may therefore get lost when writing back these images from those programs.

The formats **EMF** (Windows Enhanced Metafile), **WMF** (Windows Metafile), and **EPS** (Encapsulated PostScript) are vector drawing formats suitable to be embedded into various applications for Microsoft Windows, Mac or UNIX.

The **pwmf** format is a placable Windows metafile. This format is generally preferable to raw metafile data if the drawing will be imported as images into graphics applications or word processor documents.

The various Windows bitmap (**bmp**) formats differ in their color depth. A smaller color depth saves significant storage space, but if the color depth is too small, color information may be lost. Usually, 8 or 4 bits are a good choice. For black and white drawings, 1-bit bitmaps are sufficient. Unfortunately, many simple applications do not support all bitmap variants.

Generally, the 24 bit bitmap (**bmp24**) is most portable, but these files are generally very large. Compressed 8 and 4-bit images (**bmp4c**, **bmp8c**) are significantly smaller than the uncompressed variants, but once again one may face portability problems.

Default value:

This parameter has no default value.

Example:

Generating a GIF file of the input structure stored in example.sdf:
`mn_image -format gif ./examples/six_examples_1.sdf`

Remark:

Please note, that the resulting GIF file `six_examples_1_00001.gif` is written to the subdirectory `six_examples_1`.

-name record/name

Two different naming schemes for the image files are available. The default style `record` generates file names, which are assembled from the root name of the original file, a record count, and the standard image format suffix (`.gif`, `.png`, `.wmf`, etc.).

Alternatively, the name style constructs file names which consist of the name (property `E_NAME`) of the compound, plus the image format suffix. If the input file does not contain names, the formula is substituted. Within file names, white space is replaced by underscores.

Default value:

By default, this parameter is set to `record`.

Example:

Generating a GIF file having a filename derived from the compound name:
`mn_image -format gif -name name ./examples/six_examples_1.sdf`

Remark:

Now the resulting GIF file is named `AZT.gif` and is written to the subdirectory `six_examples_1`.

-directory <directory name>

The default image directory selection process described in the introductory section is superseded if a named target directory is specified with this option. The directory must be writable. It can optionally be automatically cleared with the **-cleardirectory** option.

Default value:

This flag is deactivated by default.

Example:

Generating a GIF file which should be stored in a given directory:
UNIX: `mn_image -format gif -directory /tmp/mn_imagedemo ./examples/six_examples.sdf`
WINDOWS: `mn_image -format gif -directory C:\tmp\mn_imagedemo ./examples/six_examples.sdf`

Example:

Now the resulting GIF files are named `six_examples_00001.gif`, etc. and are written to the subdirectory `tmp/mn_imagedemo`.

-cleardirectory

If this option is set, the target directory (either specified by the **-directory** option, or automatically chosen) is completely cleared before the image generation commences. Subdirectories will not be deleted.

Default value:

This flag is deactivated by default.

Example:

Clearing the target directory, then generating a GIF file:

```
mn_image -format gif -cleardirectory ./examples/six_examples.sdf
```

Remarks:

This option should be used with great care!

-count <n>

The parameter **-count** is able to limit the amount of processed structures. A maximum of <n> records from the input file(s) is processed. This count applies to each individual input file. Especially while handling large input files with hundreds of thousands of structures this parameter is useful to convert only the first hundred structures, if you would like to test other parameters and would not like to wait until the entire file was processed.

Default value:

No limitation (value= ∞)

Example:

Generating GIF files of the first four alkanes (from methane to butane):

```
mn_image -format gif -count 4 ./examples/alkanes1_12.sdf
```

Remark:

The **-offset** parameter can be used to position the file before the count begins and thus convert only a region of a large file.

-offset <n>

This parameter controls the entry point into each file.

Default value:

If this parameter is not set, conversion starts at the first entry (value=1).

Example:

Generating GIF files of three alkanes (decane, undecane, dodecane):

```
mn_image -format gif -offset 9 ./examples/alkanes1_12.sdf
```

Remarks:

If this parameter is used in combination with the **-count** option, sections of larger files can be processed.

-feedback

If the parameter **-feedback** is set, a dot is printed on the standard error channel for every ten completed images.

Default value:

It is not active by default.

Example:

Generating thousands of GIF files while a dot for every 10 images is plotted:

```
mn_image -format gif -feedback ./examples/maybridge.sdf
```

-antialiasing 0/1

This parameter controls whether textual content (atom symbols and annotations, header and footer lines) are rendered with antialiased fonts. It does not have any effect if the built-in fonts instead of external TrueType fonts are used.

Default value:

By default, this option is enabled.

Example:

Generating a GIF file with antialiasing support of a given input structure:

```
mn_image -format gif -antialiasing 1 ./examples/six_examples.sdf
```

-interlace 0/1

Both PNG and GIF image formats support interlacing. An interlaced image can already be displayed in lower resolution, but full size before all data has arrived. This is a useful feature in WWW environments and similar application scenarios. The disadvantage of interlacing is that it makes the images files larger. GIF images only support interlacing between lines, while PNG interlaces both between lines and columns.

Default value:

By default the parameter is set to 0.

Example:

Generating a GIF file with interlaced feature:

```
mn_image -format gif -interlace 1 ./examples/six_examples.sdf
```

-copyright <text>

If a string, such as a copyright notice, is provided for the parameter **-copyright**, this expression is embedded in the image as a hidden copyright message. It is stored in a comment field and can be extracted by viewers which are capable of displaying this kind of additional meta-information.

Default value:

This flag is deactivated by default.

Example:

Generating a GIF file containing a copyright notice:

```
mn_image -format gif -copyright "(c) Molecular Networks GmbH"
./examples/six_examples_5.sdf
```

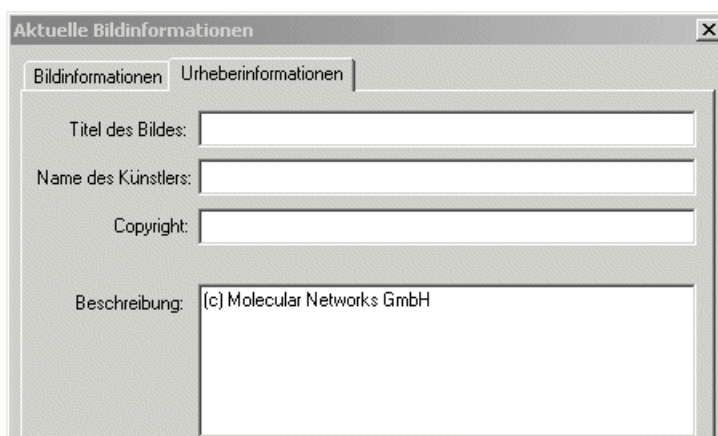
Result:

Headerline of the result file:

```
GIF87a...(c) Molecular Networks GmbH
```

Picture:

Displaying the embedded text with the picture information window of a graphics program:



Remarks:

It is also possible to use the standard Unix command strings to extract this information.

-embed none/smiles

The program supports the embedding of structure information into the images. If structure information is stored, it is possible to reclaim the molecule information without going through a lengthy and error-prone chemical OCR procedure. The smiles style stores the structure data as SMILES string.

Default value:

This flag is deactivated by default.

Example:

Generating a GIF file with embedded SMILES information:

```
mn_image -format gif -embed smiles ./examples/six_examples_5.sdf
```

Result:

Embedded SMILES string in the header of the resulted file:

```
SMILES ... O1C(\C=C\[C@@](\C=C\[C@@H](\C(=C/CC[C@H]([C@@H]1C)C)C)OC)(O)C)=O
```

Remarks:

This string can be extracted either by suitable software with the CACTVS toolkit, or simply by standard Unix tools such as grep and strings. By default, no structure code is embedded.

-metadata 0/1

If this option is set, the image will contain, in an invisible embedded data field, a DC-compatible metadata set.

Default value:

This flag is deactivated by default.

Example:

Generating a GIF file containing meta data information:

```
mn_image -format gif -metadata 1 ./examples/six_examples_5.sdf
```

Remarks:

Care must be taken when post-processing these images: many image manipulation programs do not support the handling of auxiliary data, which may therefore get lost when writing back these images from those programs.

8.5. Program features for setting the layout of the image

The following figure shows an overview of all options used for specifying the layout of the images.

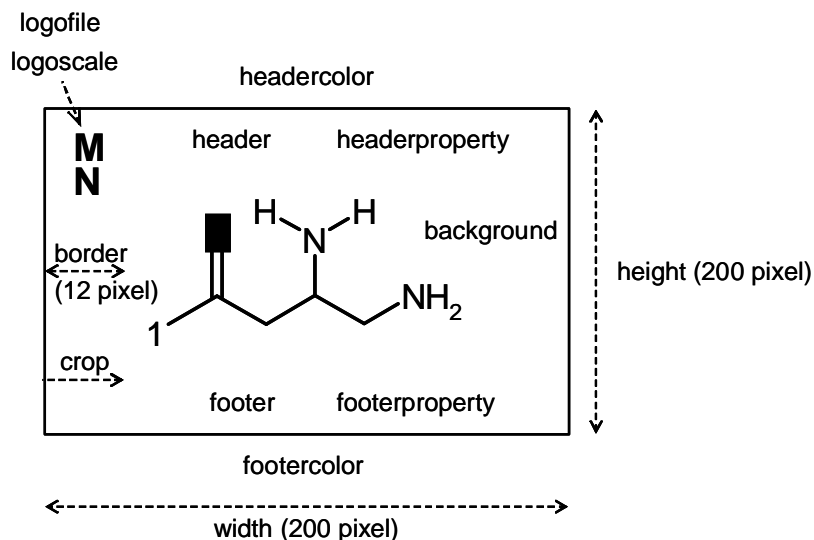


Figure 2: Overview of all options used for setting the layout of the image; in parenthesis the default values are given.

-height <pixels>

The parameter **-height** sets the height of the image in pixels. If the **-crop** option is used, this parameter determines only the height of the initial drawing area, not necessarily the height of the final image.

Default value:

The default are 200 pixels.

Example:

Generating a GIF file with a height of 300 pixels:

```
mn_image -format gif -height 300 ./examples/six_examples_5.sdf
```

Remarks:

Use the option **-width** to set the width of the image.

-width <pixels>

The parameter **-width** sets the width of the image in pixels. If the **-crop** option is used, this parameter determines only the width of the initial drawing area, not necessarily the width of the final image.

Default value:

The default are 200 pixels.

Example:

Generating a GIF file with a width of 300 pixels:

```
mn_image -format gif -width 300 ./examples/six_examples_5.sdf
```

Remarks:

Use the option **-height** to set the height of the image.

-background <colorname>/transparent

The parameter **-background** determines the background color of the image. If the special value transparent is chosen, display programs will show whatever is behind the image in those regions which are not covered by the plot or logo.

Default value:

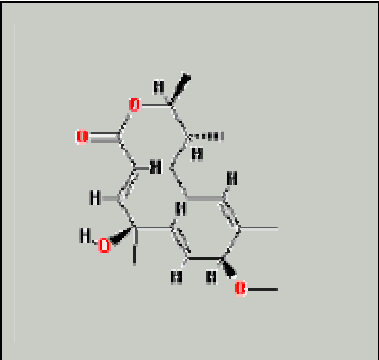
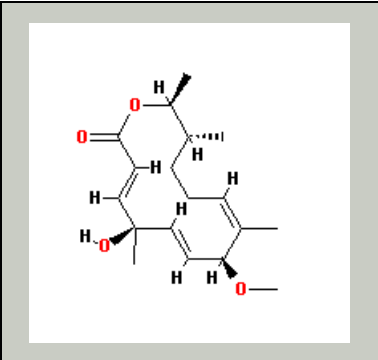
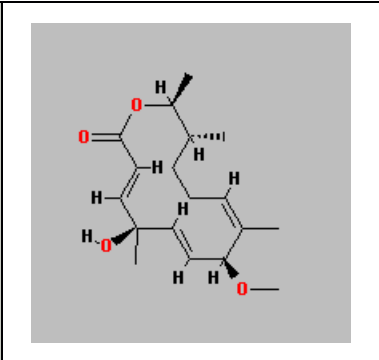
This option is deactivated by default.

Example:

Generating a GIF file with a transparent background:

```
mn_image -format gif -background transparent
./examples/six_examples_5.sdf
```

Pictures:

		
background transparent (the background color of the cell was set to grey)	background white (the background color of the cell was set to grey)	background grey (the background color of the cell was set to white)

-border <pixels>

The parameter **-border** determines the width of the border from the center of the outmost atoms of a structure which fits tightly into the display area to the outer border of the image. Note that atoms with plotted symbols require a few pixels in all directions around the atom center, so setting this parameter to very small values should be avoided.

Default value:

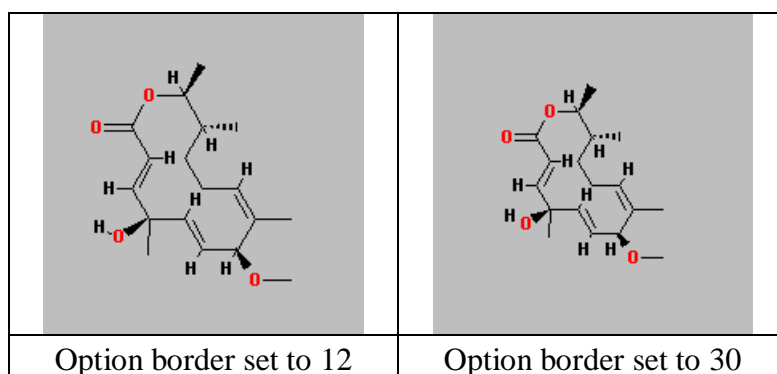
The default is 12 pixels.

Example:

Generating a GIF file with a border of 30 pixels:

```
mn_image -format gif -border 30 ./examples/six_examples_5.sdf
```

Picture:



-crop <nborderpixels>

By default, the image size is precisely defined by the **-height** and **-width** parameters (default: 200x200 pixels). The **-crop** option can be used to remove unused border space on pixel-based output format. It does not work on vector formats. In a first step, the number of continuous pixel columns and rows which contain only background pixels are determined, counting from the outsides, and these rows and columns are removed. Next, a background-colored frame of the specified **-crop** width is added. It is possible to expand the image beyond its initial size with this option, but normally images become smaller and loose their aspect ratio.

Default value:

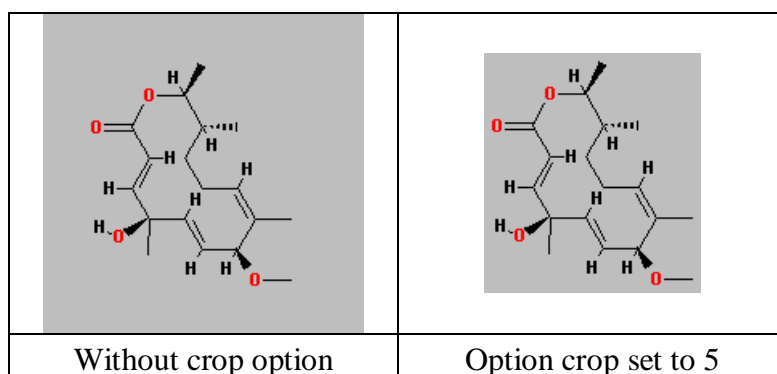
If this parameter has a negative value (the default), no cropping takes place.

Example:

Generating a GIF file with removed unused border space:

```
mn_image -format gif -crop 5 ./examples/six_examples_5.sdf
```

Picture:



-header <text>

This is a free text which is centered on the top of every image. Compound data can be automatically inserted into the header with the **-headerproperty** option.

Default value:

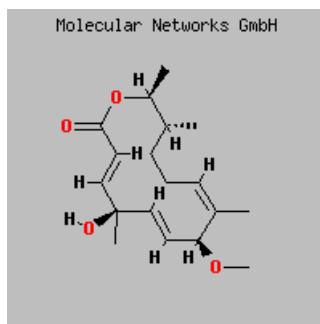
This option is deactivated by default.

Example:

The title of each image is set to "Molecular Networks GmbH":

```
mn_image -format gif -header "Molecular Networks GmbH"
./examples/six_examples_5.sdf
```

Picture:



-headerproperty <propertyname>

If no explicit header is set with the **-header** option, this option can be used to transfer the data associated with a defined property into the header field. Useful properties are for example E_NAME, E_FORMULA, E_WEIGHT, or E_SMILES. If the data is not yet present, but a method is available to compute the data from available information, it is automatically invoked. Currently, the header property must be of the ensemble property attachment type, and subfield extraction is not yet supported.

Default value:

This option is deactivated by default.

Example:

The title of each image is set to the molecular weight or the name of the compound:

```
mn_image -format gif -headerproperty E_WEIGHT
./examples/six_examples_5.sdf
mn_image -format gif -headerproperty E_NAME
./examples/six_examples.sdf
```

Picture:

<p>308.4168</p>	<p>NSC323230 Coaracine (racemic)</p>
<p>Molecular weight calculated on the fly</p>	<p>E_NAME read from input file</p>

Remarks:

Note that only properties associated with a molecular ensemble can be used, but no atom, molecule, and bond properties. In case of externally defined properties (such as SD file data fields), the corresponding internally defined name should be used. If MN.IMAGE knows about a computational method to derive the requested information from the data it read from the input file, it is not required that the data is already present. This means that you can use E_FORMULA with any file, but E_IDENT only with files which contain data which is mapped to this property, since it is not computable. Table 2 shows a list of properties which can be calculated with MN.IMAGE:

Property name	Description	UNIX	Windows
E_WEIGHT	molecular weight	yes	yes
E_FORMULA	molecular formula	yes	yes
E_NHDONORS	amount of H-donors	yes	no
E_NHACCEPTORS	amount of H-acceptors	yes	no
E_NROT BONDS	amount of rotatable bonds	yes	no
E_INPUTDATE	date the ensemble was created or read	yes	no
E_EXACT_MASS	exact molecular weight based on the monoisotopic mass	yes	no
E_COMPLEXITY	complexity value	yes	no
E_SMILES	SMILES string of the compound	yes	yes
E_HASH	64 bit hash code	yes	yes
E_HASHSY	64 bit hash code with stereo information	yes	yes

Table 2: Some properties which can be calculated by MN.IMAGE.

-headercolor <colorname>

The option **-headercolor** selects the color for the text written with the **-header** option.

Default value:

The default color is black.

Example:

The title of each image is set to the molecular weight of the compound written in red color:

```
mn_image -format gif -headerproperty E_WEIGHT -headercolor red
./examples/six_examples_5.sdf
```

-footer <text>

This is a free text which is centered on the bottom of every image. Compound data can be automatically inserted into the footer with the **-footerproperty** option.

Default value:

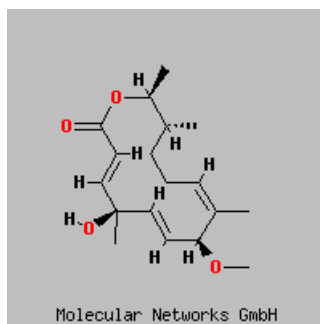
This option is deactivated by default.

Example:

The footer of each image is set to "Molecular Networks GmbH":

```
mn_image -format gif -footer "Molecular Networks GmbH"
./examples/six_examples_5.sdf
```

Picture:



-footerproperty <propertyname>

If no explicit footer is set with the **-footer** option, this option can be used to transfer the data associated with a defined property into the footer field. Useful properties are for example E_NAME, E_FORMULA, E_WEIGHT, or E_SMILES. If the data is not yet present, but a method is available to compute the data from available information, it is automatically invoked. Currently, the footer property must be of the ensemble property attachment type, and subfield extraction is not yet supported.

Default value:

This option is deactivated by default.

Example:

The footer of each image is set to the molecular weight of the compound:

```
mn_image -format gif -footerproperty E_WEIGHT
./examples/six_examples_5.sdf
```

Remarks:

Note that only properties associated with a molecular ensemble can be used, but no atom, molecule, and bond properties. In case of externally defined properties (such as SD file data fields), the corresponding internally defined name should be used. If MN.IMAGE knows about a computational method to derive the requested information from the data it read from the input file, it is not required that the data is already present. This means that you can use E_FORMULA with any file, but E_IDENT only with files which contain data which is mapped to this property, since it is not computable. Table 3 shows a list of properties which can be calculated with MN.IMAGE:

Property name	Description	UNIX	Windows
E_WEIGHT	molecular weight	yes	yes
E_FORMULA	molecular formula	yes	yes
E_NHDONORS	amount of H-donors	yes	no
E_NHACCEPTORS	amount of H-acceptors	yes	no
E_NROTBONDS	amount of rotatable bonds	yes	no
E_INPUTDATE	date the ensemble was created or read	yes	no
E_EXACT_MASS	exact molecular weight based on the monoisotopic mass	yes	no
E_COMPLEXITY	complexity value	yes	no
E_SMILES	SMILES string of the compound	yes	yes
E_HASH	64 bit hash code	yes	yes

E_HASHSY	64 hit hash code with stereo information	yes	yes
----------	--	-----	-----

Table 3: Some properties which can be calculated by MN.IMAGE.

-footercolor <colorname>

The option **-footercolor** selects the color for the text written with the **-footer** option.

Default value:

The default color is black.

Example:

The footer of each image is set to the molecular weight of the compound written in red color:

```
mn_image -format gif -footerproperty E_WEIGHT -footercolor red
./examples/six_examples_5.sdf
```

-logfile <filename>

If this is not an empty parameter, an attempt is made to read the file as GIF or PNG logo. If the image could be read, it will be inserted into the upper left corner of all produced images as a logo picture. If cropping is active, the logo will be added after cropping, so its placement will not influence the cropping process, but it could then potentially overlap atoms and bonds. Without cropping, the logo is inserted first, and other items plot over it. The logo image can be scaled by the **-logo** scale option. Logo embedding is currently not supported for the EPS and EMF formats.

Default value:

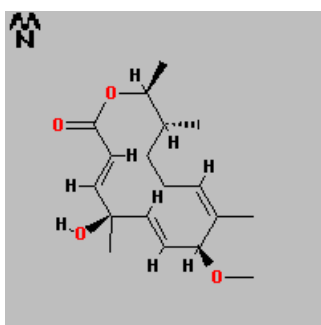
This option is deactivated by default.

Example:

Generating a GIF file with an incorporated logo image:

```
mn_image -format gif -logfile ./examples/mn_logo.gif
./examples/six_examples_5.sdf
```

Picture:



-logoscale <factor>

This factor can be used to resize a logo file specified by the **-logfile** option. The scale factor is a floating point number.

Default value:

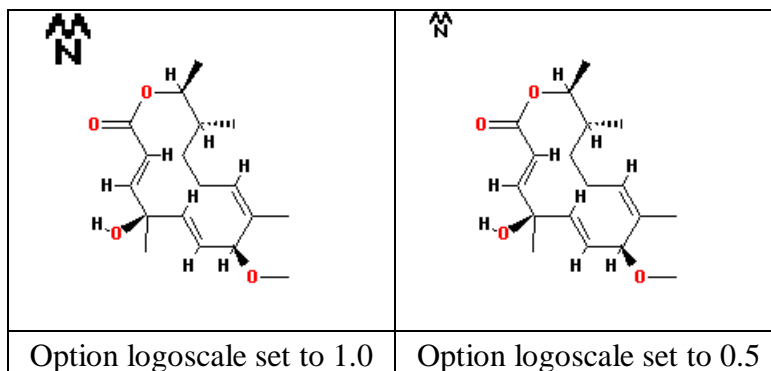
The default scaling factor is 1.0.

Example:

Generating a GIF file with an incorporated logo image scaled to half of the size:

```
mn_image -format gif -logfile ./examples/mn_logo.gif -logoscale  
0.5 ./examples/six_examples_5.sdf
```

Picture:



Remark:

Note that this option should not be used on a regular base - a cleanly rendered logo image in the intended final resolution generally has a better graphical quality than a rescaled logo.

8.6. Program features for setting the layout of molecules

-asymbol symbol/xsymbol/label/index/box/compact

The option **-asymbol** sets the display of the element symbols. By default (mode **symbol**) atoms are displayed with their element symbols (refer to the **-csymbol** and **-hsymbol** options for further methods to control the display of atoms). For normal atoms, the display types **symbol** and **xsymbol** are equivalent, but **xsymbol** will produce a more detailed text for certain types of query atoms, such as atom lists.

Alternatively, atoms can be displayed as their label (mode **label**) or atom table index (mode **index**). If the structures are read from a file encoded in a format which does not allow the storage of specific atom labelling, both **index** and **label** are equivalent and integers, beginning with 1, which indicates the position of the atom in the internal atom table. The mode **box** will suppress the display of atom symbols and use colored boxes for hetero atoms instead. The **compact** mode merges hydrogen atoms at hetero atoms (such as in NH₂) into the primary atom symbol. The layout of the structure is also taken into account, so that an HO- symbol is used if the connection point of an OH group is to the right. The hydrogen atoms which are merged into the primary symbol and their bonds are no longer displayed.

Default value:

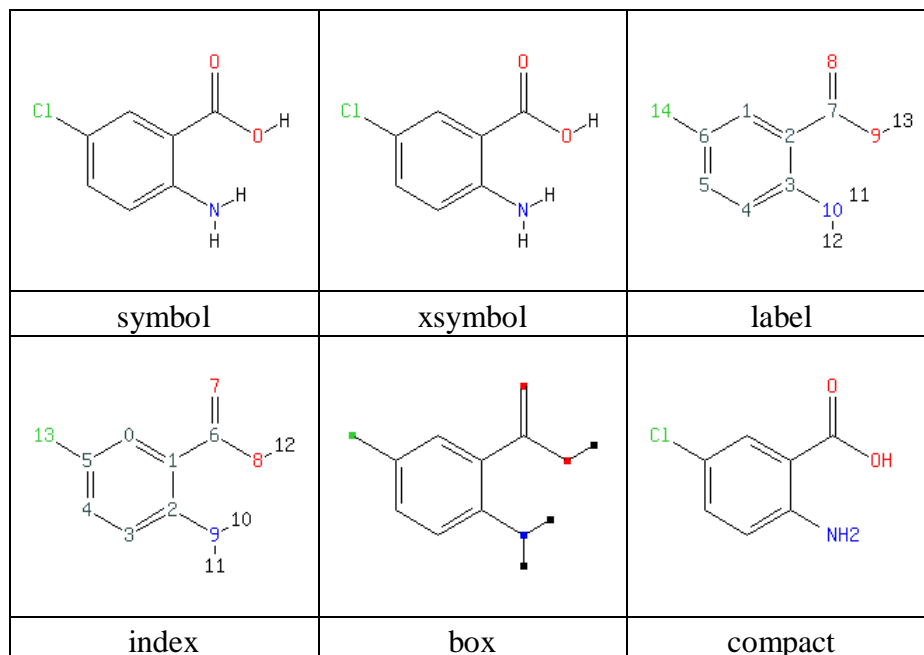
By default this option is set to **symbol**.

Example:

Generating a GIF file showing the atom symbols of each structure in compact mode:

```
mn_image -format gif -asymbol compact ./examples/asymboltest.sdf
```

Picture:



-hsymbol none/special/all

The parameter **-hsymbol** controls the rendering of the hydrogen atoms. By default (mode *special*) only hydrogen atoms which are traditionally plotted are printed, such as hydrogen

atoms on hetero atoms and aldehydes. The mode *all* will display all hydrogen atoms, whereas mode *none* suppresses them all. In contrast to carbon atoms, the bonds to suppressed hydrogen atoms also vanish.

Default value:

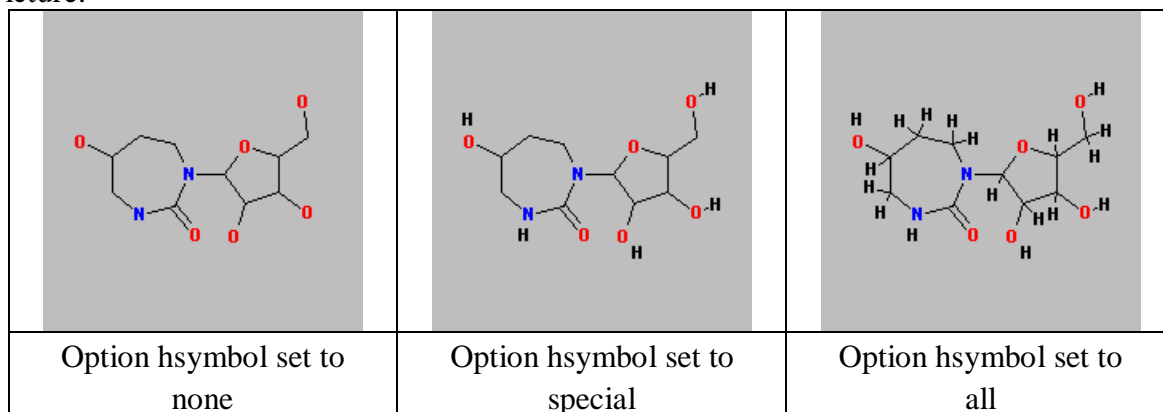
special (plot special hydrogen atoms)

Example:

Generating a GIF file without any hydrogen atom:

```
mn_image -format gif -hsymbol none ./examples/six_examples_2.sdf
```

Picture:



Remarks:

To control the rendering of carbon atoms, use the option **-csymbol**.

-csymbol none/special/all

The parameter **-csymbol** controls the display of the carbon atom symbols. By default (mode *special*) only carbon atoms which are traditionally plotted are printed, such as carbon atoms of a triple bond or with a formal charge. Other carbon atoms are displayed as nodes only. The mode *all* will display all carbon atoms as C, whereas mode *none* suppresses them all.

Default value:

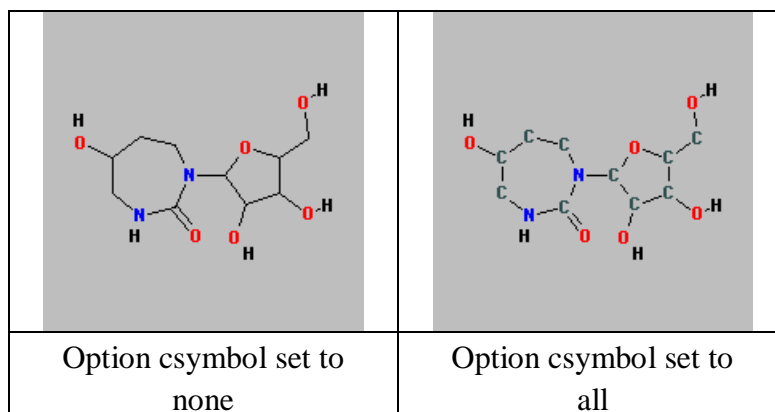
special (plot special carbon atoms)

Example:

Generating a GIF file showing all carbon atoms:

```
mn_image -format gif -csymbol all ./examples/six_examples_2.sdf
```

Picture:



Remarks:

To control the rendering of hydrogen atoms, use the option **-hsymbol**. This parameter is ignored if the global **-asymbol** style is label, index or box.

-atomcolor <colorname>/type

Select a global color for all atom symbols. If the default color **type** is chosen, the color of atom symbols is determined individually. If the input data contains color information, it is used. Otherwise, a standard element-specific coloring scheme is applied. For hydrogen atoms, the hydrogen color specified with the **-hcolor** option will override both a global atom color and an individual atom color, if it is not an empty string.

Default value:

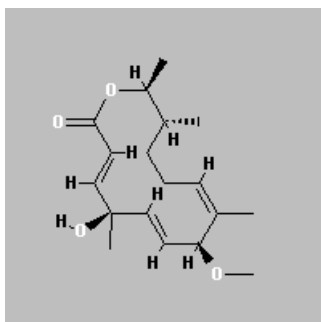
By default this option is set to "type".

Example:

Generating a GIF file with white colored atom symbols for non-hydrogen atoms:

```
mn_image -format gif -atomcolor white -background grey
./examples/six_examples_5.sdf
```

Picture:



-hcolor <colorname>

The option **-hcolor** specifies an override color for hydrogen atoms. If the color name is not an empty string, it overrides both the global atom color, and an individual hydrogen color taken from file or the standard element color table.

Default value:

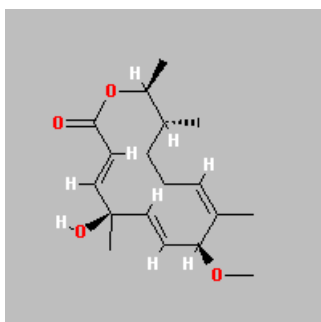
This flag is deactivated by default.

Example:

Generating a GIF file with white colored atom symbols for hydrogen atoms:

```
mn_image -format gif -hcolor white -background grey
./examples/six_examples_5.sdf
```

Picture:



-bondcolor <colorname>/split

The parameter **-bondcolor** allows the selection of a global color for bond lines. In contrast to atom colors, individual bond colors are currently not supported. The special value **split** splits the bond into two halves. Each half bond is colored in the same color as its associated atom symbol, with the exception of carbon atoms. In case of half bonds to carbon atoms, the split color is either black or white, automatically selected according to the image background.

Default value:

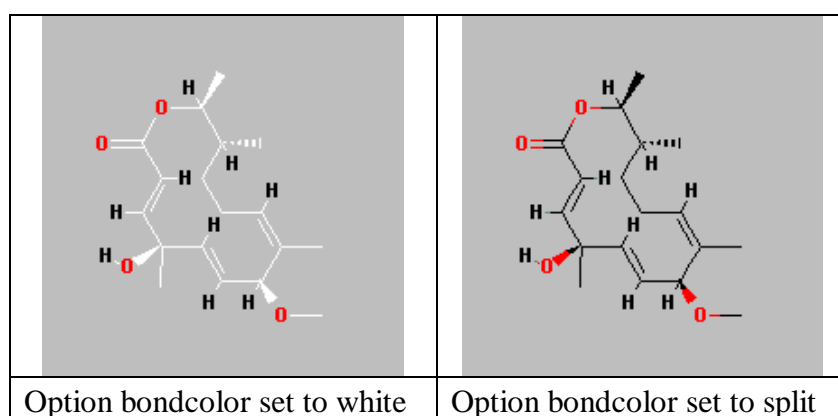
By default, the color of bonds is either set to black or white.

Example:

Generating a GIF file showing white colored bonds:

```
mn_image -format gif -bondcolor white -background grey
./examples/six_examples_5.sdf
```

Picture:



-symbolfontsize <pts>

The option **-symbolfontsize** specifies the size of element symbols or labels in the structure displays. The default is -1, meaning that the program should automatically choose a suitable standard font size, which may be scaled down if a structure is shrunk (see **-bondscale** option). If this value is set to 0, no symbols will be printed. Hetero atoms are then marked by small (colored, if in color mode) squares. The desired point size can be a floating point number.

Default value:

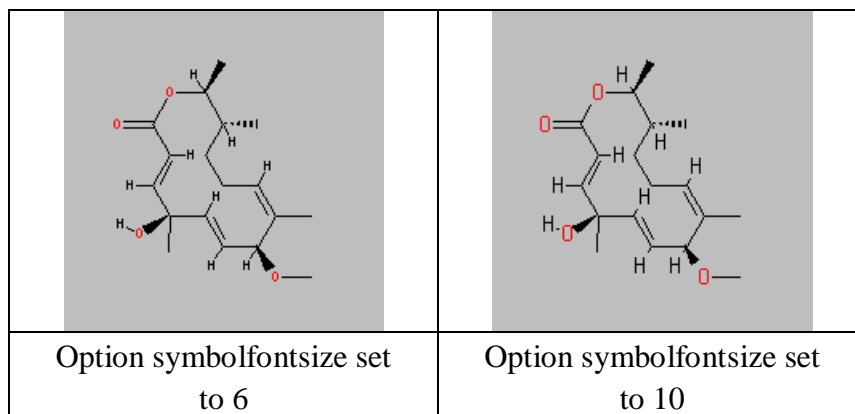
By default the program chooses suitable font sizes for atomic symbols automatically.

Example:

Generating a GIF file showing the atom symbols of each structure in font size 10 pts:

```
mn_image -format gif -symbolfontsize 10
./examples/six_examples_5.sdf
```

Picture:



Remark:

Note that the number of available font sizes is limited if you use the built-in fonts instead of external TrueType fonts. The program will choose the closest built-in font (from the list 6pt, 7pt, 8pt, 10pt) in this case.

-annotationfontsize <pts>

The option **-annotationfontsize** controls the font size for atom annotations, such as formal charges or isotope labels. If the value is negative (the default), the font size is automatically derived from the element symbol size. If it is set to zero, no atom annotations are printed.

Default value:

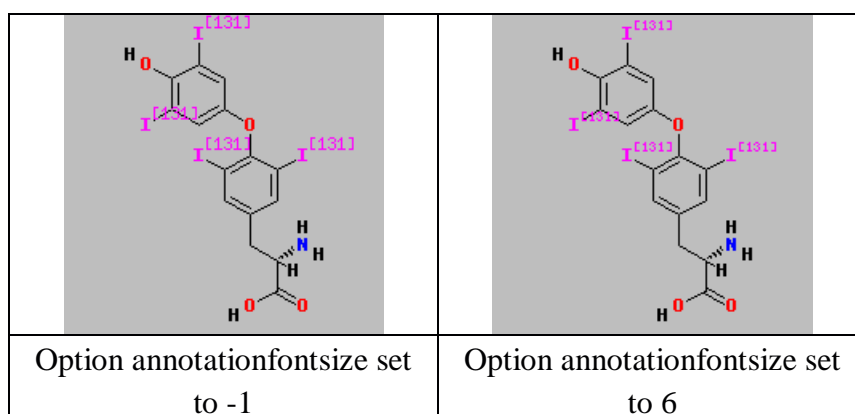
By default the font size is set to -1, meaning automatically adjustment.

Example:

Generating a GIF file with a given annotation font size of 6 pts:

```
mn_image -format gif -annotationfontsize 6
./examples/thyroxin_i131.sdf
```

Picture:



Remark:

Note that the number of available font sizes is limited if you use the built-in fonts instead of external TrueType fonts. The program will choose the closest built-in font (from the list 6pt, 7pt, 8pt, 10pt) in this case. The desired point size may be a floating point number.

-font fontfile

The parameter **-font** sets the font file. By default, this option is set to the empty string and thus a small collection of built-in fonts are used for the image rendering. However, instead of the built-in set, any TrueType font can be used as a replacement. Using TrueType fonts has several advantages. Only these fonts can be scaled to an arbitrary point size, and they can be rendered with anti-aliasing (see option **-antialiasing**). The font file can either be a full path to a .ttf file, or the simple body of the file name (with or without the suffix). If the font file is not found as a fully qualified file, a search path is traversed, which includes the fonts/ subdirectory in the standard installation. This directory is therefore the preferred place to store font files. So, an Arial Narrow font in that directory could both be specified as /usr/local/lib/cactvs/fonts/arialn.ttf or simply arialn.

Default value:

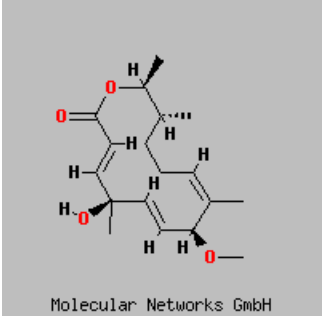
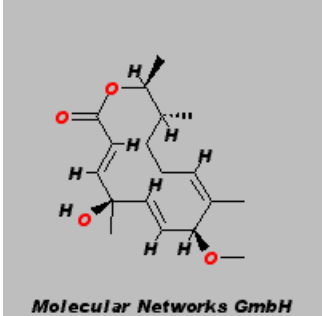
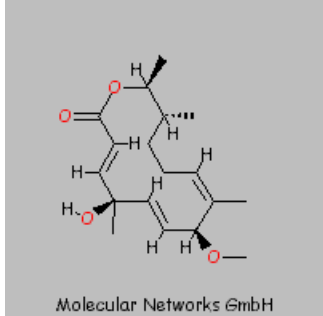
By default built-in fonts are used.

Example:

Generating a GIF file using the font "Comic":

```
mn_image -format gif -font comic.ttf -footer "Molecular Networks  
GmbH" ./examples/six_examples_5.sdf
```

Picture:

		
Without setting the font option	Option font set to "Arial Bold Italics"	Option font set to "Comic Sans MS"

-bondscale <n>

The option **-bondscale** specifies the number of bonds in x-direction which are fitted into a structure box without re-scaling. The default value is 8. Structures with an x extent of 12 standard bond lengths fit into the structure box tightly on the left and on the right. Smaller structures will be centered and have extra space to the left and right. Larger structures are shrunk so that they still fit into their boxes. This mechanism ensures that structures of normal (up to the scale parameter) will all be drawn with the same bond lengths and thus appear graphically pleasing, without allowing larger compounds to break out of their boxes or cutting off part of their graph. Note that re-scaling may still occur if the structure exceeds the height of the structure box. Re-scaling is always uniform in order to avoid distortions. Setting this parameter to a smaller value lets small structures make better use of the area of their boxes, but increases the likelihood of enforced size adjustment.

Default value:

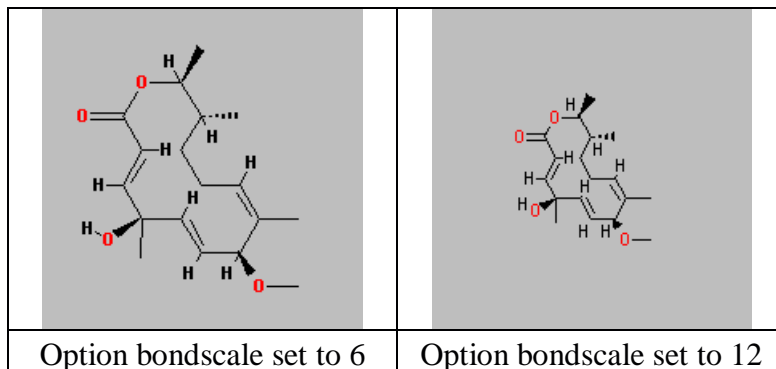
By default the option bondscale is set to 8.

Example:

Generating a GIF file with a bondscale of 12:

```
mn_image -format gif -bondscale 12 ./examples/six_examples_5.sdf
```

Picture:



-bead 0/1

If the option **-bead** is set, certain generic atom types (R, polymer, search classes) are depicted as two-color circles with the shape of beads. This feature is intended to be used in conjunction with polymer-support chemistry.

Default value:

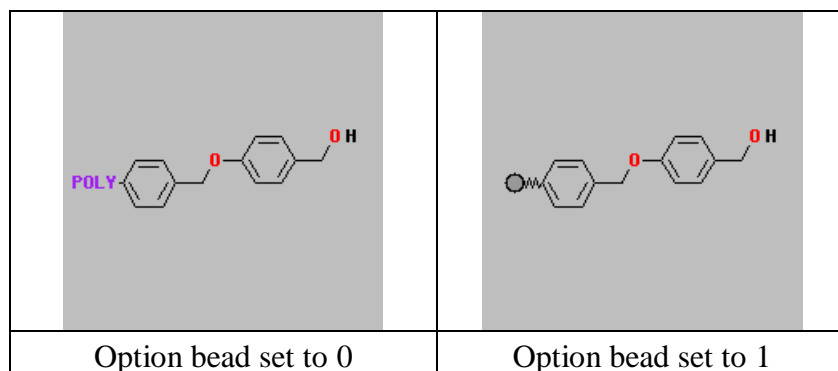
This option is active by default.

Example:

Generating a GIF file displaying the atom "Poly" as circle:

```
mn_image -format gif -bead 1 -border 20 ./examples/polymer.sdf
```

Picture:



-showcharge <0/1>

The parameter **-showcharge** controls the plotting of charge symbols. If this flag is set to 0, atomic charge symbols are not plotted.

Default value:

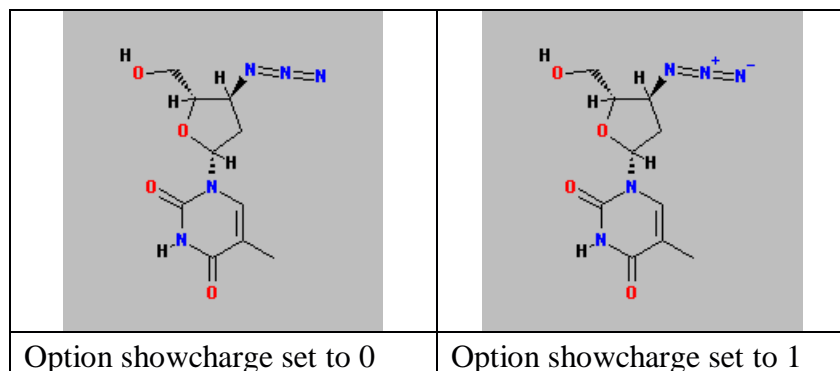
By default, charge symbols are plotted (value 1).

Example:

Generating a GIF file showing no charge symbols:

```
mn_image -format gif -showcharge 0 ./examples/six_examples_1.sdf
```

Picture:



-showisotope <0/1>

The parameter **-showisotope** controls the plotting of the nucleonic number. If this flag is set, isotopically labeled atoms are annotated with their nucleonic number. Heavy hydrogen atoms are displayed as D or T. If the flag is unset, isotopic information is suppressed.

Default value:

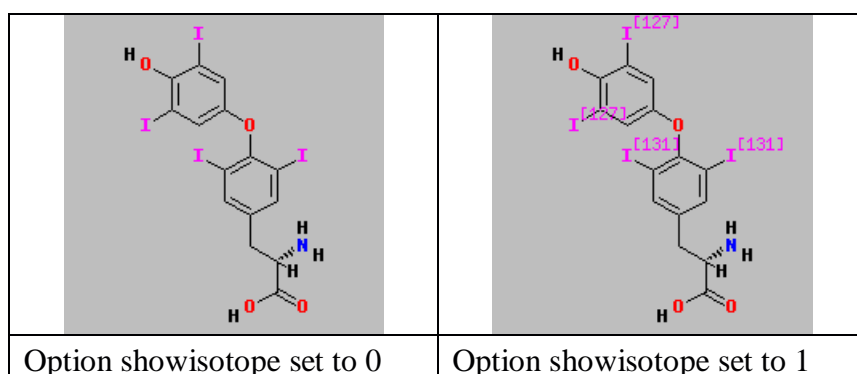
By default, nucleonic symbols are plotted (value 1).

Example:

Generating a GIF file showing isotope information:

```
mn_image -format gif -showisotope 1
./examples/thyroxin_i127_131.sdf
```

Picture:



-showradical <0/1>

The parameter **-showradical** controls the plotting of radical centers.

Default value:

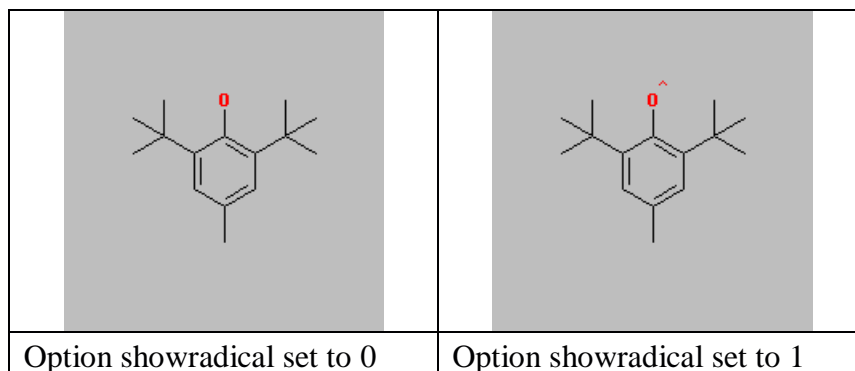
By default, radical centers are plotted (value 1).

Example:

Generating a GIF file showing no radical centers:

```
mn_image -format gif -showradical 0 ./examples/bht_radical.mol
```

Picture:



-showstereo <0/1>

The parameter **-showstereo** controls the plotting of stereo descriptors. If this flag is set to 0, no stereo descriptors are plotted. Note that this flag has no influence on the display of wedge bonds (see **-wedges** and **-dashes** options to control their appearance). It only applies to atomic stereo descriptors such as CIP R or S, which might be present in the input file. These descriptors are not computed if not explicitly present in the input data. They are only plotted if read from file.

Default value:

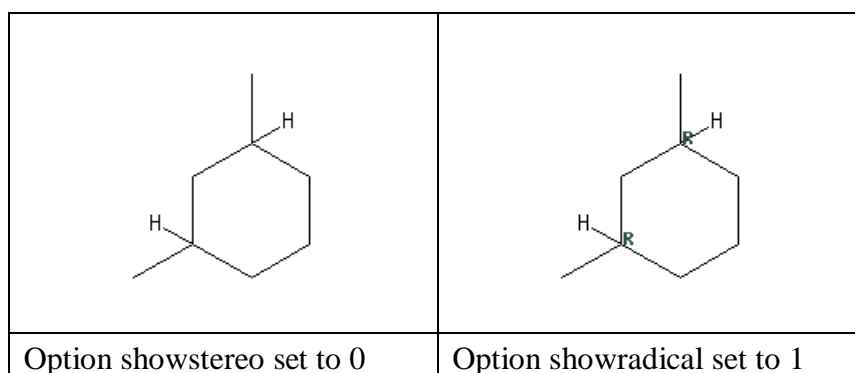
By default, no stereo descriptors are plotted (value 0).

Example:

Generating a GIF file showing stereo descriptors:

```
mn_image -format gif -showstereo 1 ./examples/stereo.sdf
```

Picture:



-showstereoh <0/1>

The parameter **-showstereoh** controls the plotting of hydrogen atoms at stereo centers. If this flag is set, hydrogen atoms at stereo centers which are not linked via a wedge bond are explicitly drawn in order to obtain an unambiguous stereocenter display. Hydrogen atoms linked via a wedge bond are always drawn regardless of the value of this flag.

Default value:

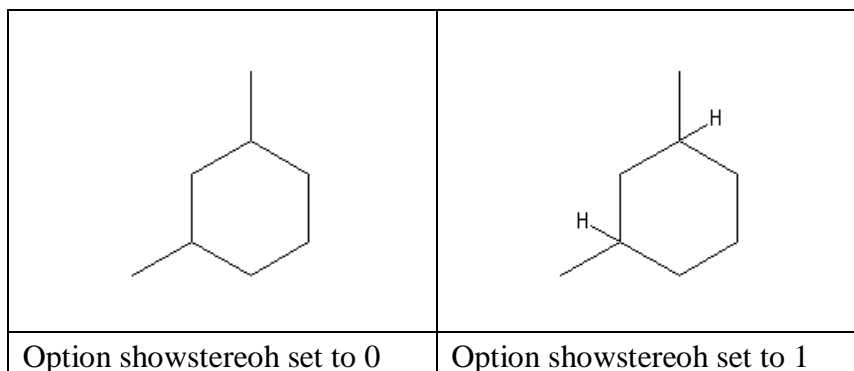
By default, hydrogen atoms at stereo centers are plotted (value 1).

Example:

Generating a GIF file showing no stereo descriptors at hydrogen atoms:

```
mn_image -format gif -showstereoh 0 ./examples/stereo.sdf
```

Picture:



-wedges 0/1

The option **-wedges** controls whether wedge bonds are displayed as such or not.

Default value:

This flag is activated by default (value=1).

Example:

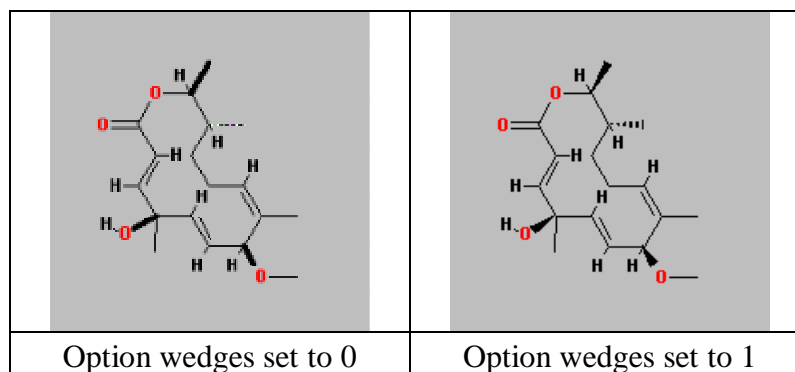
Generating a GIF file without any attributes on wedge bonds:

```
mn_image -format gif -wedges 0 ./examples/six_examples_5.sdf
```

Remarks:

If this parameter is set to 0, but the **-dashes** parameter is still switched on, full wedges will be printed as a bold bond and dashed wedges as simple dashed bond. If the **-dashes** option is also inactive, the display of bond stereo attributes is completely suppressed.

Pictures:



-dashes 0/1

The option **-dashes** controls whether the dash attribute on wedges and simple bond lines are displayed as such or not.

Default value:

This flag is activated by default (value=1).

Example:

Generating a GIF file without any attributes on wedge or dash bonds:

```
mn_image -format gif -dashes 0 -wedges 0
./examples/six_examples_5.sdf
```

Remarks:

In order to avoid wrong display of stereochemistry, it should only set to 0 in combination with a **-wedges 0** option. Resetting this option alone, without also resetting **-showstereo**, is not useful.

8.7. Program features of general usage

-version

If this flag is set, the version and licensing information is printed.

Default value:

This flag is deactivated by default.

Example:

Showing the program version:

```
mn_image -version
```

9. Extended Features only available for the UNIX operating systems

Input files can be processed in compressed or gzip-ed form without prior unpacking. The input file name arguments may each be a local file, an URL (http, ftp, gopher, file) or an email message file containing the structure data in the main body or as one or more attachments. URL retrieval and compression can be combined.

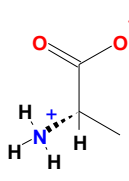
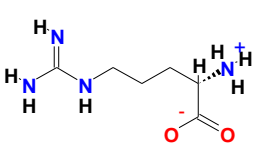
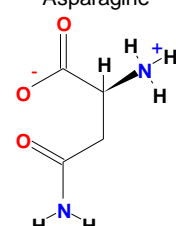
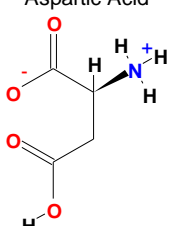
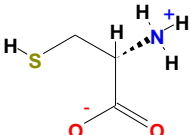
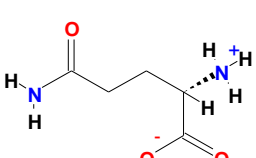
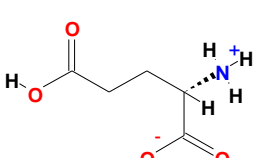
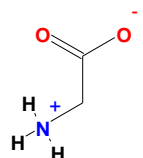
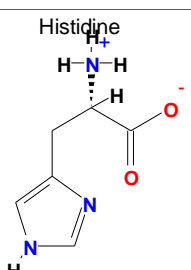
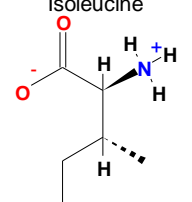
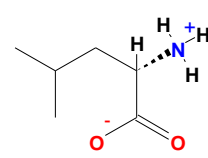
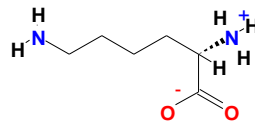
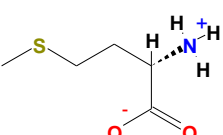
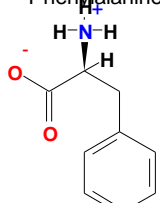
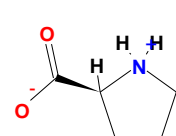
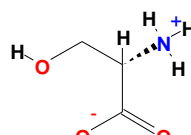
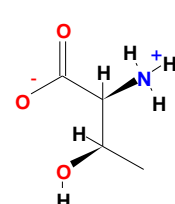
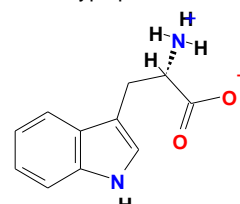
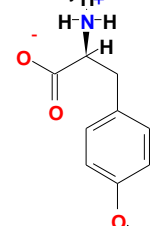
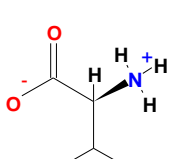
The output file name can also be an anonymous ftp URL.

10. Application examples

10.1. Table of 20 natural amino acids

The 20 natural amino acids (20as.sdf) were depicted using the MN.IMAGE program. In the header line of each image the molecule name was printed.

Natural Amino Acids

Alanine 	Arginine 	Asparagine 	Aspartic Acid 
Cysteine 	Glutamine 	Glutamic Acid 	Glycine 
Histidine 	Isoleucine 	Leucine 	Lysine 
Methionine 	Phenylalanine 	Proline 	Serine 
Threonine 	Tryptophan 	Tyrosine 	Valine 

11.Frequently Asked Questions (FAQ)

12.Error messages

13.Known problems and limitations

14. Technical Support

The MN.IMAGE Web Site

If you have problems while running MN.IMAGE please have a look at the Support- and FAQ web site of MN.IMAGE. The pages are available at <http://www.mol-net.de>

Reporting Problems

If your problem is not listed in these web pages please report it to the MN.IMAGE team at Molecular Networks. Please make sure to provide us with all important data for replicating your problem on our machines. Therefore please use the report form on the next page.

Updates

If you have licensed the program MN.IMAGE with maintenance you will automatically receive updates every time a new release is launched.

Contact information

Distribution and Maintenance for MN.IMAGE is handled by Molecular Networks Computerchemie, Erlangen, Germany.

Molecular Networks GmbH
Computerchemie
Nägelsbachstraße 25
91052 Erlangen
Germany

e-Mail: support@mol-net.de

Tel. +49 9131/815668

Fax +49 9131/815669

15.Report Form

In case of problems occurring during installation or running MN.IMAGE, please complete the following form and send it or fax it to

Molecular Networks GmbH Computerchemie
Nägelsbachstraße 25
91052 Erlangen
Germany
FAX: +49-(0)9131-815669

User:

MN.IMAGE program and version number (mn_image -version):

Command line to run MN.IMAGE:

Error and warning messages by MN.IMAGE:

System messages:

Short description:

Please include the input file and output file generated by MN.IMAGE on a 3½" diskette written in MS/DOS format or send an e-mail to support@mol-net.de attaching these files. These files will help us to analyze your problems. All data will be treated confidentially.

16.Index

inputfile
 20as.sdf 37
 alkanes1_12.sdf 15
 asymboltest.sdf 26
 bht_radical.mol 33
 maybridge.sdf 16
 polymer.sdf 32
 six_examples.sdf 10, 14, 15, 16, 21
 six_examples_1.sdf 14, 32
 six_examples_2.sdf 27
 six_examples_5.sdf 16, 17, 18, 19, 20,
 21, 22, 23, 24, 25, 28, 29, 31, 32, 35
 stereo.sdf 34
 thyroxin_i127_131.sdf 33
 thyroxin_i131.sdf 30

option
 annotationfontsize 30
 antialiasing 16
 asymbol 26
 atomcolor 28
 background 19
 bead 32
 bondcolor 29
 bondscale 31
 border 19
 cleardirectory 15
 copyright 16
 count 15
 crop 20
 csymbol 27
 dashes 35
 directory 14
 embed 17
 feedback 15
 font 31
 footer 22
 footercolor 24
 footerproperty 23
 format 13
 hcolor 28
 header 20
 headercolor 22
 headerproperty 21
 height 18
 hsymbol 26
 interlace 16
 logfile 24
 logoscale 24
 metadata 17
 name 14
 offset 15
 showcharge 32
 showisotope 33
 showradical 33
 showstereo 34
 showstereoh 34
 symbolfontsize 29
 version 36
 wedges 35
 width 18