mindmap 2

The Visual Application Assembly Environment

User's Guide

mindmap Software Corporation
2520 Mission College Boulevard
Santa Clara, CA 95054
USA

LIMITED WARRANTY

mindmap Software warrants the software media to be free of defects in workmanship for a period of 30 days from purchase. During this period mindmap Software will replace at no charge any such media returned to mindmap Software, postage prepaid. This service is mindmap's sole liability under this warranty.

DISCLAIMER

LICENSE FEES FOR THE SOFTWARE DO NOT INCLUDE ANY CONSIDERATION FOR ASSUMPTION OF RISK BY MINDMAP SOFTWARE OR ITS LICENSOR, AND MINDMAP SOFTWARE AND ITS LICENSOR DISCLAIM ANY AND ALL LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR OPERATION OR INABILITY TO USE THE SOFTWARE, OR ARISING FROM THE NEGLIGENCE OF MINDMAP SOFTWARE AND ITS LICENSOR, OR THEIR EMPLOYEES, OFFICERS, DIRECTORS, CONSULTANTS OR DEALERS, EVEN IF ANY OF THESE PARTIES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. FURTHERMORE, LICENSEE INDEMNIFIES AND AGREES TO HOLD MINDMAP SOFTWARE AND ITS LICENSOR HARMLESS FROM SUCH CLAIMS. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY THE LICENSEE. THE WARRANTIES EXPRESSED IN THIS LICENSE ARE THE ONLY WARRANTIES MADE BY MINDMAP SOFTWARE AND ITS LICENSOR, AND ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE. THIS WARRANTY GIVES YOU SPECIFIED LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF WARRANTIES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

We welcome your comments and suggestions about this manual. Please send your comments via email to support@mindmap-software.com or write to us at the above address.

Part number 01-100004-0001

Acknowledgments

Thanks to all who contributed to the success of this document. Special thanks to Frank 'The Raver' Ihringer, Martin 'Mr. Detail' Opitz, Felix 'Mr. Arctanh' Opitz, comical Martin Schlichenmayer, the suave Alexander Friedrich and reliable Thomas Schumann.

If you don't know where you are going, all paths will lead you there.

About the Book

The manuscript for this manual was prepared in Microsoft Word for Windows 7.0b simultaneously in three countries on two continents. Formatting was done in this same great program, although we did, at times, feel like we were pushing the limits.

Drawings were done in Adobe Photoshop using the Wacom Digitizer.

The body has been set in Century Schoolbook. The headings are set in Humanist BT. Notations are set in Comic Sans MS.

Trademarks

Contents

Chapter 1 What is mindmap ?

Background

What mindmap can do

First and foremost, mindmap is an environment in which applications can be developed. Since computers ultimately execute machine instructions, a development environment must eventually generate such instructions. Over the brief history of computers, major advances have been accomplished in isolating the developer from the low level of machine code. Each new level of isolation has been accompanied with more abstraction. At each new level the developer has to deal less and less with machine based concepts. In the very early days, it used to be AND and OR switches. Then came registers and machine instructions. These were followed by variables, conditionals, loops, and so forth.

mindmap is a member of the newest development level - graphical development. It is a language, albeit a graphical language.

The price we have paid in general for the increase in power has been a loss of control. The more complex the entities that are manipulated, the less influence one has regarding the details. All in all, more can be accomplished, but at the expense of being limited by the scope of the environment.

Let's briefly look at the power issue. What mindmap offers is twofold. Persons competent in development, in general, are now able to perform most tasks much easier, faster and will be less prone to errors. Persons not having a background in development, can now build their own applications, without having to resort to 'agents' to do the work for them.

mindmap is an environment in which applications are actually assembled. Existing building blocks (components) are used, ar-

ranged and connected, such that the result is an application. mindmap is not intended to be used as an environment in which building blocks are created.

Traditional development environments offer a language, a debugging facility and a variety of supplemental tools. The language is supplied in the form of a compiler or interpreter. The developer enters statements into an editor and has the statements translated - either up front (compiler) or at runtime (interpreter). The statements input reflect a certain, and hopefully intended, procedural flow of things. Therefore, the developer must conceive of the flow (semantics) and then mold it into the language (syntax). In order to easily locate and eliminate syntactic and semantic errors in the code, a debugger is most often supplied. This helps the developer step through the application and view isolated states. Finally, most environments offer various other tools that ease the general process of building applications (linkers, profilers, etc.).

Here, mindmap differs dramatically from the traditional approach to development, in that it shifts the emphasis of development. The syntax of mindmap is very limited. Statements are not actually keyed in. They are selected from various lists, which are displayed depending on the situation. It is not possible to create a syntactically invalid statement. mindmap is also non-procedural, so that the concept of mapping the flow to a series of statements is also completely different. The process is much more comparable to painting a picture. What You See is What You Get (WYSIWYG) best describes it.

The assembler of an application selects a building block, defines various attributes and then links it to other blocks. Think of it as a grown-up version of an erector set or Lego® blocks. First, one selects the type of block from an assortment of different blocks (wheels, plates, blocks, windows, etc.). Next, a decision is made regarding the attributes of the selected block type (color, size, number of panes, etc.). Finally, the block is combined with other blocks (knobs are plugged into holes, parts are screwed together, etc.).

Obviously, the types of applications that can be built with the environment is a function of the available components, their attributes and the ability to connect the components. This is fundamentally a question of expandability. mindmap ships with a large assortment of components, having many attributes, and numerous methods of being combined with one another. Never-

theless, the system offers diverse means of incorporating new components.

What we assume you know

The first major assumption is that you know how to handle MS Windows. We assume that you are familiar with MS-Windows concepts and associated conventions. This implies that you know how to deal with either a mouse and/or the equivalent keyboard entries.

If you will be using mindmap in conjunction with databases, we assume that you are knowledgeable of general database concepts . This means that you are comfortable with the concepts of fields, records, and tables. If the underlying database engine you are using supports SQL (Structured Query Language), and it becomes necessary to execute an action by employing SQL, we also assume that you are either familiar with the SQL language or have access to other sources of information about SQL.

If you will be using mindmap to construct multimedia applications and will be using MCI commands (Media Control Interface), we make the assumption that you are either already knowledgeable or have access to other sources of information on MCI.

It will be helpful if you have some experience in application development. This does not mean that we expect you to be able to write code, but having experience in programming usually means that you:

- know how to decompose complex tasks into smaller, more manageable pieces;
- are accustomed to first analyze and then design, before you jump to the actual realization;
- have been exposed to the concepts of variables, loops, subroutines, etc.

Please do not misunderstand this point. It is absolutely not necessary to be a programmer in order to be able to build applications with mindmap. It simply makes it easier, in some cases, in that general programming concepts can be utilized. We must

point out, though, that sometimes programming experience is a hindrance, in that it becomes difficult to think in non-procedural terms.

Throughout this manual and your use of mindmap, we will discuss various options or alternatives in the use and configuration of the components. You are highly encouraged to try the alternatives discussed. Only by experimenting, will you truly come to realize the depth of possibilities offered in mindmap. We look forward to your comments, suggestions and general feedback.

Hardware and Software Requirements

mindmap appreciates powerful hardware, but it does not give up on systems that are not state-of-the-art. The faster, bigger and better your system is, the faster your mindmap applications will run. Being a little more specific…

The more RAM you have, the better it is. mindmap will function on a 4MB system, although you might not enjoy the performance. We suggest a minimum of 8MB - obviously the more, the better.

Additional RAM is more important to mindmap than a faster microprocessor. Obviously, the faster the processor, the faster mindmap will execute. mindmap will function on a 386SX, although we would suggest a 486 as a minimum processor.

Since mindmap itself, and most applications built with it, are very user interface oriented, and thus utilize graphics extensively, we suggest a fast graphics card. mindmap will function in 4-bit, 640x480 pixels graphic environments, but we again suggest at least 256 colors and a fast graphics card.

Since MS Windows is a virtual memory operating system, swapping from memory to the hard disk and back is quite common. mindmap and applications built with mindmap are generally compact in size, but, nonetheless, must often be swapped to a hard disk. Once mindmap has been installed, it does not make special demands concerning the hard disk. The faster the drive, the more free space it has, and the less fragmented it is,

the better. mindmap consumes approximately 5MB of hard disk space after installation.

Since mindmap is an MS-Windows application it does require one of the currently available MS-Windows environments. mindmap currently supports the 16-bit interface in MS-Windows, so it will run on the following versions of MS-Windows:

- MS-Windows 3.11
- MS-Windows for Workgroups
- MS Windows 95

mindmap will run as a 16-bit application on Windows 95. mindmap will also support the common dialog boxes and long file names. mindmap currently does not support MS-Windows for Pen.

If you plan on using the supplied database interface, you will require Microsoft's ODBC (Open Database Connection). mindmap installs a version on your system, but please take caution here; there might be a conflict with other ODBC applications on your system. mindmap supports the 16- and 32-bit implementations of ODBC. In order to have access to some of the more sophisticated features (scrollable cursors, etc.), you must use ODBC 2.x.

If you use videos (AVI files) or other multimedia features in mindmap, it will be necessary to have Video for Windows and/or any other required drivers installed (Sound card etc.). Video for Windows is included in many packages. Please refer to the vendor of your hardware for additional information.

If you have installed other software products, you might have opted to also install some supplied graphic filters. In this case, they will be recognized by mindmap and offered, along with the standard mindmap filters, when you attempt to import graphic files. These filters are denoted with an asterisk in front of the file type in the corresponding dialog box, while the mindmap filters appear without the asterisk. This applies to 16-bit versions of import filters which support the Lotus/Intel/ Microsoft standard.

In any event, it is important that you consider not only your system and its configuration in the use of mindmap, but more

importantly, the system configuration of the systems on which your completed application will be deployed.

Document Conventions

Through out the documentation, we have made it a point to supply you with as many Tips, Tricks, Hints and Warnings as possible. The intention is to offer as much help as possible where we expect you to need it. These inserts are easily recognized, as they are set off from the rest of the text as shown with this example.

Throughout this document and in the online Help, some conventions have been made to assist in their ease of use.

- Menu Options:

  All selections from the menu are in a different typeface (Humanist Bold). In order to more easily distinguish the various levels, each entry is separated with a vertical line. An example is:

  File | Open

- Keyboard Entries:

  Whenever this documentation refers to a keyboard entry, the string is set in a different typeface (`Courier`). An example might be

  `F1`

  When you encounter a string set in this manner, you are expected to key in the appropriate characters.

- Buttons:

  Generally, buttons are not displayed in their graphic representation. We normally display just the button label, but we use a different font. An example would be

  OK

  When you read a string in this setting, please press the associated button in mindmap.

Suggested Reading

If you will be dealing with databases in your applications, we strongly recommend that you make yourself familiar with SQL. Although mindmap attempts to isolate you from SQL as much as possible, it might become necessary to generate more complex

statements using SQL. A number of books and general introductory literature is available. Some of the titles we suggest include:

Author	Title
Bowman, et al	Practical SQL Handbook, Addison-Wesley, 1996
Gruber	Understanding SQL, Sybex 1990
Taylor	SQL for Dummies, IDG 1995

If you plan on creating applications that include multimedia features (sound, video, animation, etc.), you will be using Microsoft's MCI (Media Control Interface) language. The scope of this language is basically determined by Microsoft, but it has provisions for vendor specific extensions. Beside referring to the appropriate Microsoft literature, it might also be necessary to read the information provided by the vendor of your additional hardware/software.

Technically, mindmap is a visual programming language. It includes a syntax — which is graphical and does not permit you to create syntactically incorrect statements. It has a domain — which is limited to building a rather broad scope of applications under MS Windows. Since it does not do away with underlying concepts commonly found in application building (loops, variables, subroutines, etc.), we recommend that you eventually devote some time to these concepts. If you have a programming background, these concepts are more than familiar to you. If you are a novice, you might run into some roadblocks, while building applications. They most likely will be of a more conceptual nature. It is therefore recommended that you spend time at least browsing through some fundamental literature dealing with these basic concepts.

mindmap directs a major focus towards the user interface. Building appealing and functional user interfaces is part science and part art.

The scientific part is dealt with in the following representative publications:

Author	Title
Aaron Marcus, Nick Smilonich, Lynne Thompson	The Cross-GUI Handbook; Addison-Wesley, 1995
Susan L. Fowler, Victor R. Stanwick	The GUI Style Guide; Academic Press, 1995

The artistic side can be found in books such as:

Author	Title
Ray Kristof, Amy Satran	Interactivity by Design; Adobe Press, 1995

Chapter 2 General Overview

Introduction

mindmap is supplied in the form of 3.5" diskettes, a CD, or you have downloaded it off the Internet. In any case, the procedure for installing mindmap is similar. The setup process will assist you in creating a directory into which the files will be copied. During the procedure you will be asked to key in your serial number. Either you will be entering a serial number for a demo version, or you will enter a key for a valid license. In both cases, all associated components will be stamped with this serial number.

In the installation process, mindmap must copy some files into the Windows directory. (We regret having to do this, but this is required by the vendors of some of the components we include. We do not consider this good style, but we cannot avoid it.) The first step is fairly straightforward and not much can go wrong. After completing this segment, you are asked whether you intend to install additional modules. These modules are additional mindmap components, some of which require the presence of third-party engines such as Lotus Notes, Wang's OPEN/image, etc. These modules are not part of the standard scope of mindmap and must be procured in addition to mindmap. Depending on the module, these may be obtained from mindmap or other third parties.

Once you have completed the installation process, a program group is created and various icons are included. Along with the actual program icon, you will see other icons representing README files and other supporting items. In the case of Windows 95, these entries are found on the Programs entry on the taskbar.

After the successful creation of the program group, you will be informed of the successful installation of mindmap. You are now ready to begin using mindmap.

The first time you start mindmap, you will be asked in which language you wish to use mindmap. Select the desired language. (Do not worry if you select the wrong language or need to change it in the future. mindmap allows for this.) mindmap will create a file in the mindmap directory called MINDMAP.INI. This file acts as a registry and contains numerous settings and user preferences for mindmap. mindmap does not make any entries in the WIN.INI or SYSTEM.INI files, since we also consider this to be poor style. Isolating all entries in the MINDMAP.INI file facilitates a very clean deinstallation, should you ever desire to do this.

Contents of Diskettes/CD

You can copy the contents of the media to your hard disk and install from the hard disk. Please note that you are required to create a directory for each diskette included in your package. If you choose to do this, it is suggested that you create a directory called \MM and directories called \MM\DISK1, \MM\DISK2, etc.

The diskettes or CD contain all files necessary to run mindmap. Additional modules offering access to various third-party engines (mail systems, OCR, fulltext, etc.) can be purchased from mindmap or another vendor, or check our Internet site for the latest offerings. All third-party engines must be purchased from a source other than mindmap. Licensing of all third-party software is the responsibility of the user.

The files contained on the media are compressed and can only be installed if you use the SETUP.EXE on the first diskette (or on the CD).

The contents of the supplied media can be found in a list on the first disk (3.5") or on the CD in a file called FILES.TXT. You can view or print this ASCII file using your word processor or any other text editor.

Installation Steps

The following description will assume you are installing from a CD-ROM. The process is actually quite similar if you are installing from diskettes, except for the added juggling of multiple disks:

Insert the CD-ROM and execute the program SETUP.EXE. This can either be accomplished by using the Windows File-Manager and double-clicking on the file or via the menu option **File | Run**. In the case of Win95, click the Start button on the

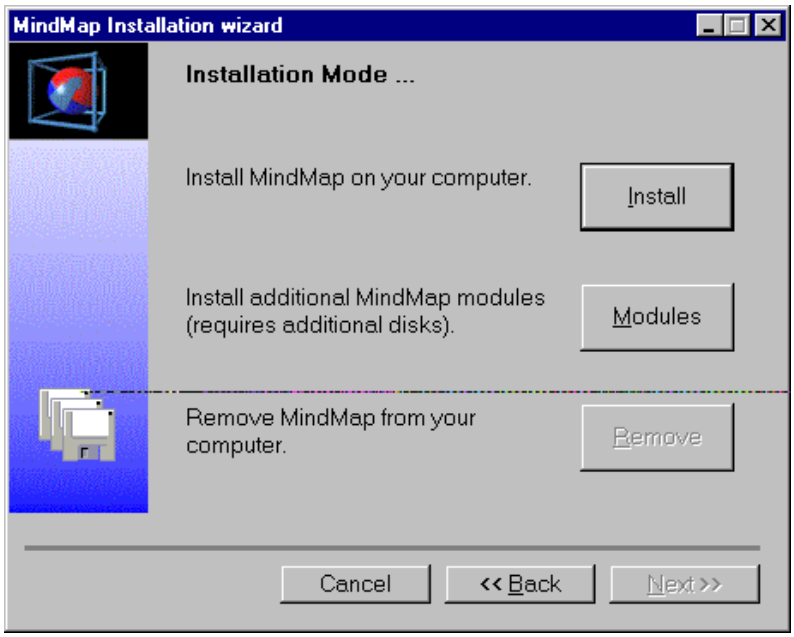
taskbar and then on the Run… option. Then browse to the SETUP.EXE file on the disk. Alternatively, you can use the Explorer to locate and subsequently execute the file.

SETUP will begin loading some of the files and then greet you with the following screen:

Installation Step 1

Clicking on the Cancel button will terminate the Setup procedure. Assuming that you wish to continue installing mindmap, click on the Next button.

This will take you to the following dialog box. Here, you are asked to select between (i) installing mindmap itself, (ii) installing mindmap modules, or (iii) removing a previously installed mindmap version.

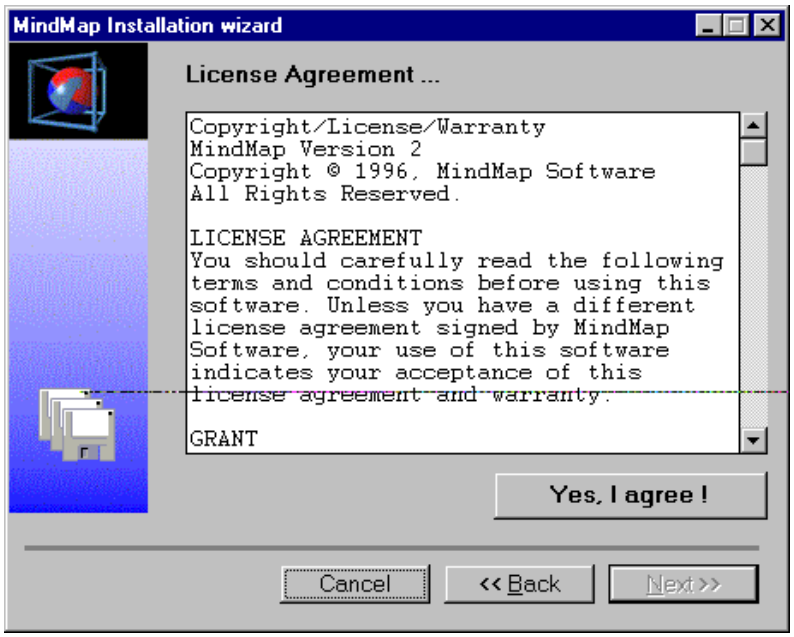


Installation Step 2

The Remove option will be grayed unless the Setup procedure has been started from the SETUP directory of a previous installation of mindmap.

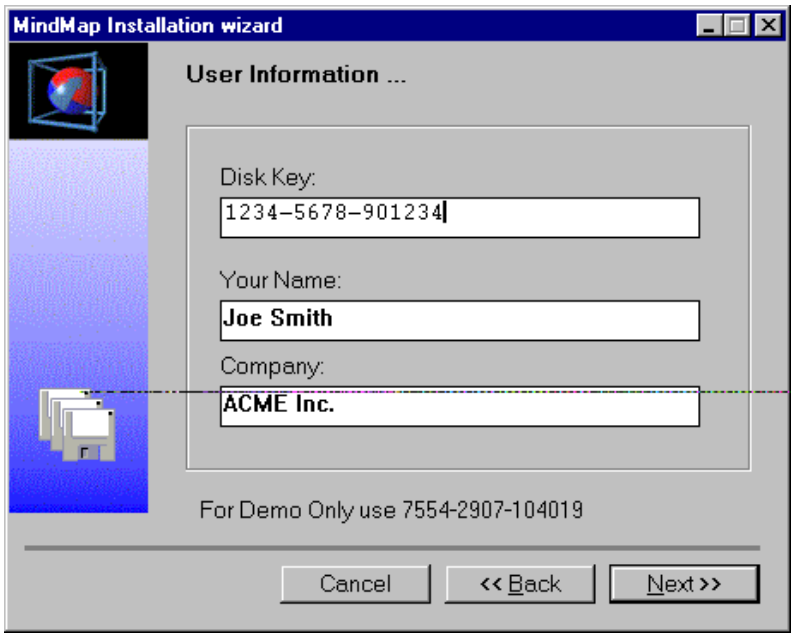
Selecting the Install option will lead to the next dialog box.

Here the current license agreement will be presented to you. Please read it carefully. If you do not wish to comply, please discontinue the installation process and promptly return the product and license to where you purchased it. If you do wish to comply, please signal this by accepting and you will be taken to the next step of the installation.



Installation Step 3

This dialog box requires you to enter the mindmap serial number, along with your name and the name of your organization.



Installation Step 4

The disk key (serial number) is an essential entry - without it, mindmap will not install properly. The disk key may be found in as many as three different locations. A label is attached to the CD-ROM package (or disk 1, if on floppies), the first page of the printed documentation (if supplied with your version), and on the Registration Card (again, if supplied.) The disk key will also be your means of access to all future offerings, including support, updates, etc. Please keep it confidential and secure.

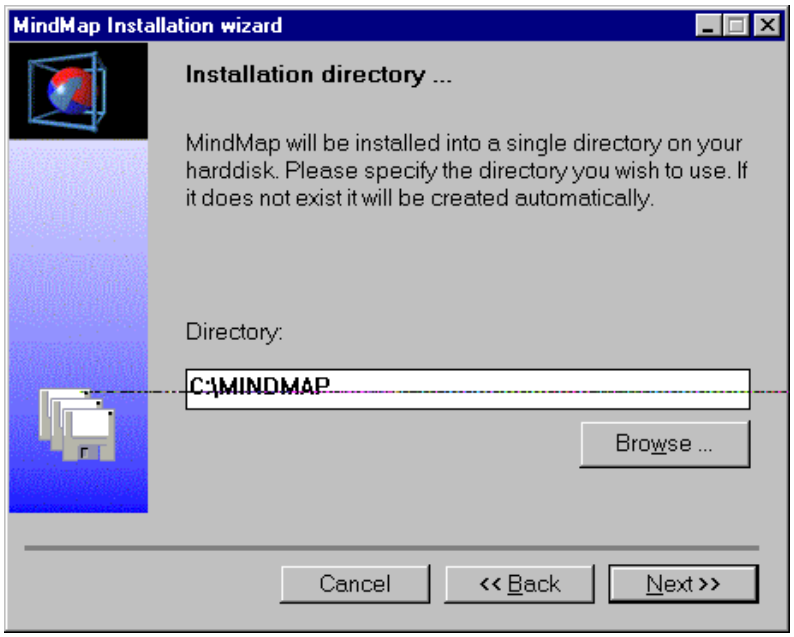
Next, enter your name and the name of your organization.

If you are installing a Demo Only version of mindmap, please use the demo disk key supplied as above or, in some cases, the default disk key displayed in the dialog box.

Under certain circumstances, you might receive the serial number when downloading the software from the Internet. In this case, please enter the key supplied.

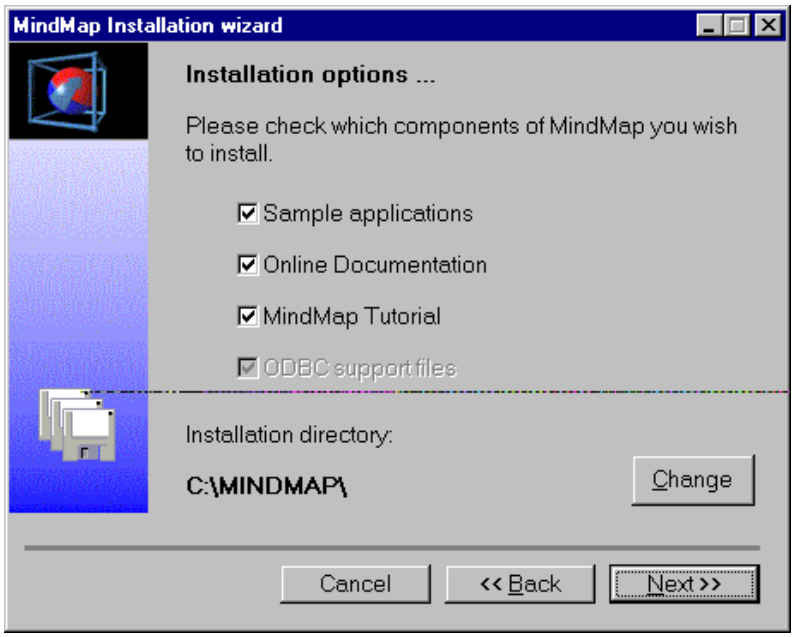
Before copying the files to the hard disk, mindmap will ask for the name of a directory. (If you previously installed mindmap, the Setup routine will detect the appropriate directory and

suggest installing the new version in the same directory.) It will default to C:\MINDMAP, but feel free to enter any directory you choose. Alternatively, you can use the Browse button to navigate to a desired directory that already exists on your hard disk.



Installation Step 5

mindmap will now ask you to make some necessary selections regarding support files.



Installation Step 6

If you decide to install the sample applications, two directories underneath the specified directory (the default would be C:\MINDMAP\SAMPLES and C:\MINDMAP\BASIC) will be created and various mindmap example applications, as well as a step-by-step tutorial, will be copied.

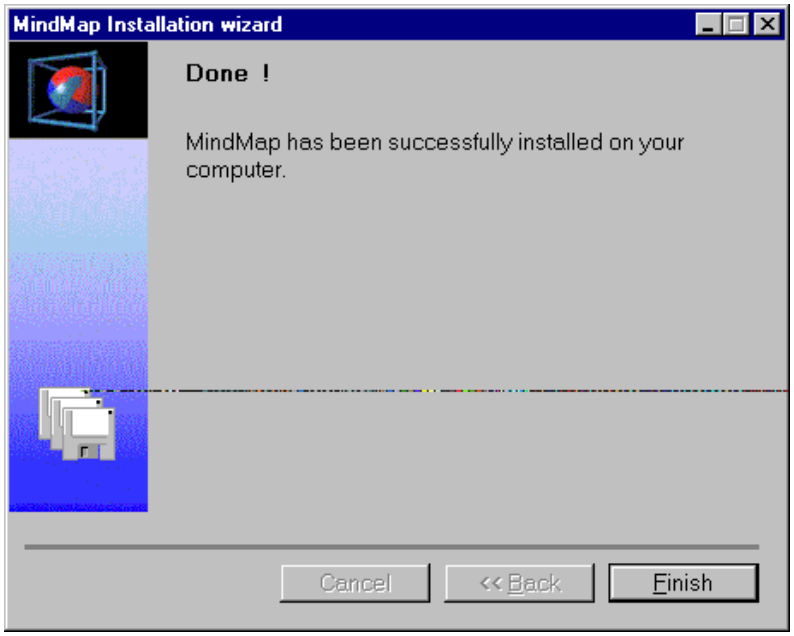
All documentation is available in generic PDF (Adobe Acrobat Reader) format on the CD in the directory \DOCU. To read and optionally print the documentation, please install the Adobe Acrobat Reader from the directory \ACROREAD on the CD.

Finally, you can also choose to install ODBC (Open Database Connectivity). You must install ODBC components if you intend to install either of the sample files or the tutorial, because some of these files rely on the existence of ODBC data sources. Please refer to your Microsoft documentation for further information regarding various ODBC settings, procedures and options.

SETUP will now begin to copy the selected files to the specified directory. If you are installing from the CD-ROM, the process can be left unattended for a few minutes. If you are working

with diskettes, you will be prompted to swap diskettes every few minutes.

The final screen will inform you that the installation process has been successfully completed. (You might also be able to briefly see various screen activity showing the creation of a program group.)



Installation Step 7

End of the installation: A new Program Manager Group

Contents of the Program Group

The following files (represented by the appropriate icons) are available in the Program Group:

- MINDMAP.EXE (the actual development environment)
- mindmap Help (the Help system)
- SETUP.EXE (mindmap Setup routine)
- mindmap Tutorial
- ODBC Administration
- What's New (lists changes since the previous version)
- The list of sample files is included in the file README.WRI in the root directory of your CD.

mindmap Settings

As mentioned above, mindmap does not make any entries in common system resources, such as WIN.INI, SYSTEM.INI, etc. mindmap keeps its settings in its own INI file. This file -

MINDMAP.INI - is not created at the time of installation. It is created at the moment you first normally terminate mindmap. Erasing the MINDMAP.INI file, after it has been created, will place mindmap back in its original unused state. It will not keep mindmap from operating.

Deinstalling mindmap

If you should ever need to deinstall mindmap, the process is actually quite simple. You can either choose to use the SETUP routine to remove mindmap or you can simply delete the files from the hard disk.

If you wish to use the SETUP program, launch the SETUP.EXE in the mindmap directory and navigate until you encounter the dialog box which offers the option to remove mindmap.

The Uninstall screen

Click on the Next button to proceed.

Last dialog box before *mindmap* is removed

You will be warned, so that you do not inadvertently remove mindmap. Clicking on the Yes button will then cause the files associated with mindmap to be erased from the hard disk.

If you wish to manually remove mindmap, locate the directory in which you previously installed mindmap. Now delete all files contained in the directory. Please remember that we could not avoid copying some files into the WINDOWS directory. All of these files are related to the ODBC environment. Since ODBC files may be a shared resource (used by a variety of other applications), we cannot remove them.

If you erase only the MINDMAP.INI file, mindmap acts as though it has just been installed. When you start mindmap again, you will be prompted to enter the preferred language support.

What can go wrong

Well, there are a number of possible causes for 'misconduct'. In principal, there are four different classes of errors:

1. Media faults
   We go through extensive checks for error-free media, but errors still might slip through. This type of error usually leads to unreadable media.

   Please contact us immediately, so that we can supply you with a new set of media.

   If you have downloaded mindmap from the Internet, you may have errors from this process. Please download it again or copy to a new set of media for your installation.
2. Inconsistencies on your system
   A typical situation might be that there is not enough space on your hard disk for mindmap to be installed.

In this specific case, the remedy is obvious. You must free up enough disk space. In any other case, please contact us so that we can help you overcome these difficulties.

3. Errors on our part
   Obviously, we are not aware of any errors in the setup routine. This is not to say there aren't any, though. Please refer to the README file for the latest information. If you believe there may be an error on our part, please contact us so that we may resolve the situation.

4. User errors
   This is a tough call. If you have followed the instructions in this manual, as well as those presented on your screen, there is not much that can go wrong. If you do have a problem, again, please contact us.

   Our goal is to provide you with a fully functional installation of mindmap.

Chapter 3 Menu Bar

Use of the mindmap Menu Bar

Before we begin to explain the mindmap menu bar, let us explain the essential commands and functions in a Windows menu. Windows is a graphical user interface (GUI) that can be used comfortably with a mouse. Nevertheless, you still have the option of using the keyboard of your computer for many functions. For example, when you are writing, it might be quicker to use a key combination for accessing the appropriate command. After starting mindmap, you can access the mindmap menu commands by pressing the **ALT** key and the underlined letter. For example, **ALT+F** opens the menu of File. Cursor left or cursor right opens the menu of the immediately adjacent item.

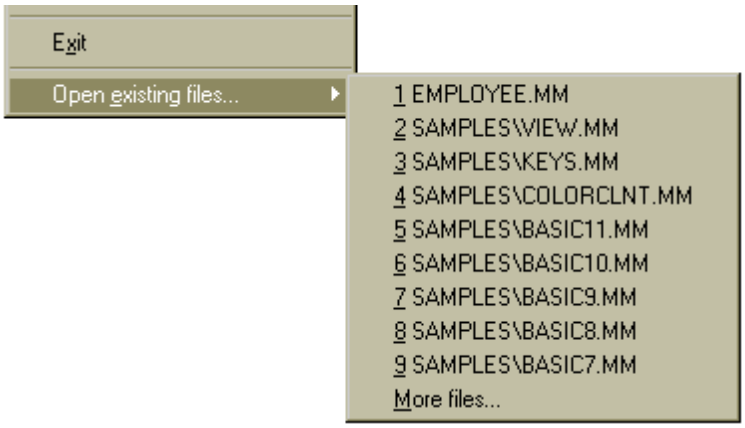
Typical mindmap menu

Pressing one of the underlined letters in the menu takes you directly to the next command line; e.g., **ALT+F+O** shows the **File | Open** dialog box. The three dots behind a command inform you that there is a branch to a dialog box where the necessary actions can be done. For example, **ALT+F+O** shows a list of files that can be opened.

Some of the commands in the menus have a key combination at the end of the line. These shortcut keys branch to the command directly. Pressing **CTRL+N** shows the New File dialog box.

The triangle ▸ at the end of the line of a menu shows that there is a fly-out menu that will give you an additional selection for the chosen command. For example, the command Open existing files gives a list of mindmap files you have already

saved. (This list of existing files is stored in the MINDMAP.INI file.)



Example of a fly-out menu

The command **File | Open existing files** is only available if you have previously saved a mindmap application.

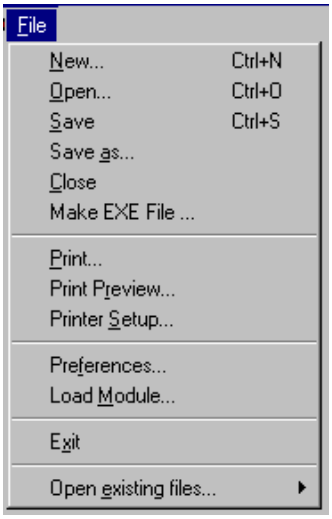
A grayed-out font in a menu indicates that this command is not available at the moment. For example, the commands Cut, Paste and Delete have a gray font as long as you have not selected a component. Selecting a component turns the gray font into a black one and gives you the option to cut, paste or delete the component.

The ✓ sign at the beginning of a command line acts as a toggle and shows that a special attribute is either on or off. Clicking on a command, such as **View | Toolbox** in the menu View, shows or hides the toolbox, so the ✓ sign is on or off.

You can close a menu selection by pressing the **Esc** key.

Please note, as in the example above, that some commands are only available when a component other than a page is selected.

File Menu



File Menu

File | New

This command opens a dialog box with the choice of either Application or Printer layout.

By choosing Application, you can open a new application. The name of the actual application is shown at the top of your window in the caption bar. It should be noted that the default name (as assigned by mindmap, Application 1) will be used here until you have first saved the application, thereby defining a real name for the application.

The Printer layout allows you to define the appearance of the documentation you can produce for your application. mindmap is shipped with a collection of layouts, which can be used as-is or as templates for your own personalized layouts.

mindmap is shipped with various standard printer layouts that are used if you print documentation of your application. The standard printer layouts are stored in the file STANDARD.MMP. You can change any layout or create your own printer layouts.

➤ For more information, please refer to page 381.

Although you can have multiple mindmap applications open on your system, you can not have more than one instance of mindmap, itself, running at one time.

You can open multiple mindmap applications at the same time. The Windows menu shows the different mindmap applications you have open and allows you to switch between them.

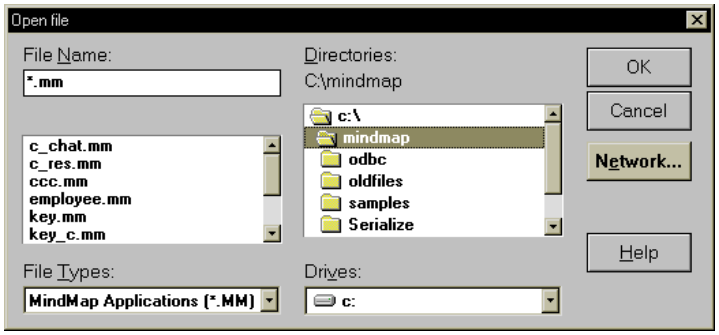
If you click on the size symbol of your mindmap applications, you can size and arrange them on your screen. You can also use the **Window | Tile** or **Window | Cascade** commands.

File | Open

This command opens a dialog box that permits you to open an existing mindmap application. mindmap shows a list of files in its working directory, normally C:\MINDMAP. When you save a mindmap application to another directory or sub-directory, the

working directory in MINDMAP.INI will be adjusted. This means that mindmap always tries to load an application from that directory in which you most recently saved a mindmap file. Note that mindmap only remembers the directory of the last file saved.

A list offers a choice of different File types: mindmap application files (*.MM), Printer Layouts (*.MMP), Backup Files (*.B0*) and all files (*.*). The selection of mindmap application files is the default.



This dialog box is used to open an existing *mindmap* application

mindmap will default to *.MM. If you saved your mindmap applications without the MM extension, you must explicitly type in the file extension or select All files *.* as the file type.

You can change the Drive: from where you want to access files or you can change the directory with a double-click on the symbol in the Directories: list.

Select the desired file with a double-click on the file name, or by selecting the file in the list and clicking the OK-button.

File | Save

If you have not previously saved the application you are working on, mindmap will prompt you for a file name. Any valid DOS file name (8 and 3 convention) is acceptable. It is suggested that you use the standard mindmap extension (*.MM) for applications you wish to edit in the future. If you are running Windows 95, you may also use the long file name feature.

This command saves the open file under the same name. The first time you save a file, a backup copy will be created with the extension *.B00. mindmap initially creates up to three backups of every file. They are named *.B00, *.B01 and *.B02 and represent the last three versions created by use of the Save command. Every time you save an application, a new backup is created and the oldest backup is overwritten. The oldest version is then stored in the *.B02 file. You can load these files into mindmap and you are then able to see the progress of your application development. If the system abnormally aborts and you cannot access the MM version of your application, you can revert to a backup copy.

You can choose if you want no backup copies at all or if you would like to have automatic backup copies created by mindmap. This is set in the **File | Preferences** menu.

File | Save as

If you want to rename an open file, you can use the command Save as. This option is also important for creating files other than mindmap application files. Since any file (with the exception of .EXE files) created by mindmap is technically identical in its structure, only the extension will be a clue to its content and the manner in which you want it to be processed. If you save an application as an *.MMP file, it will be used as a printer template. Saving the same file as *.MM will permit you to subsequently edit and run it.

We strongly recommend using the suggested file name extensions.

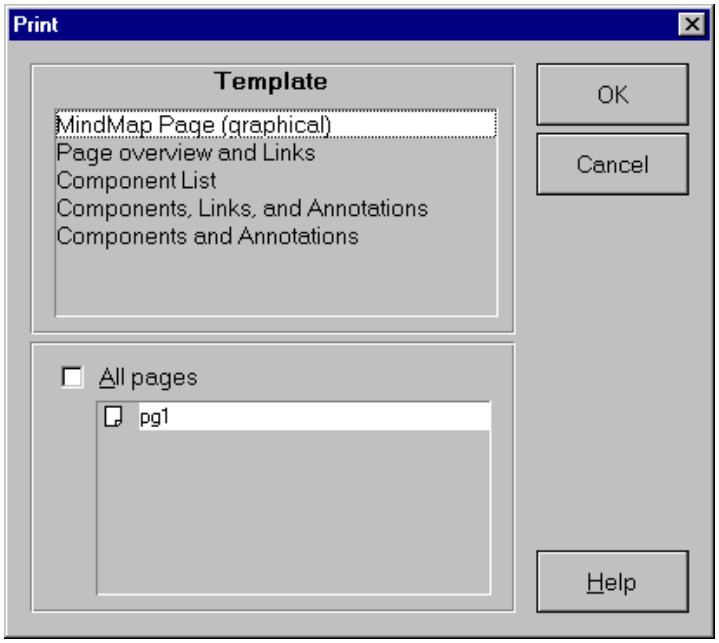
File | Close

An open application can be closed with this command. It has the same functionality as the Windows close button. mindmap does not directly close applications that have been changed. There is always a warning and you are asked: 'Do you want to save the changes of (Application)?'. This warning gives you the option to save or not; that is to keep changes or to reject undesirable changes made since you last saved the file.

File | Print

The mindmap Print function allows you to look at five different layouts. It helps you analyze your applications and to create application documentation. The standard layouts are as follows:

- mindmap Page (graphical);
- Page Overview and Links;
- Component List;
- Components, Links, and Annotations;
- and Components and Annotations.



Select how to print an application

mindmap Page (graphical)

The option mindmap Page (graphical) opens a dialog box that specifies details of the standard printer. You can define the Print area ('All' or 'Pages from: to:'), the 'Print quality' or you can 'Print to file'. If you choose the option to only print specific pages, you must select the desired pages from the list. The printout shows the components on the pages.

Page Overview and Links

With this option, you can print the layout of the pages in a mindmap application. You will see the desired page with all components on it and with the component names. In addition, the printout will list the links you have defined for each component. Links that have been marked to be inactive are indicated with a ✗ sign.

Before you print the Page Overview and Links, you can choose if you want to print all pages or only specific pages.

Object List

Object List opens the same dialog box as printing a 'mindmap Page (graphical)' and 'Page Overview and Links'. This printout shows a page oriented list with all of the components in an application. Again, you can choose if you want to print all pages or specific pages.

Objects, Links and Annotations

This selection opens the same dialog box as the previous options. The printout shows a page-oriented list of the components in an application. In addition, you will see the links and annotations of each component. Again, you can choose if you want to print all pages or only specific pages.

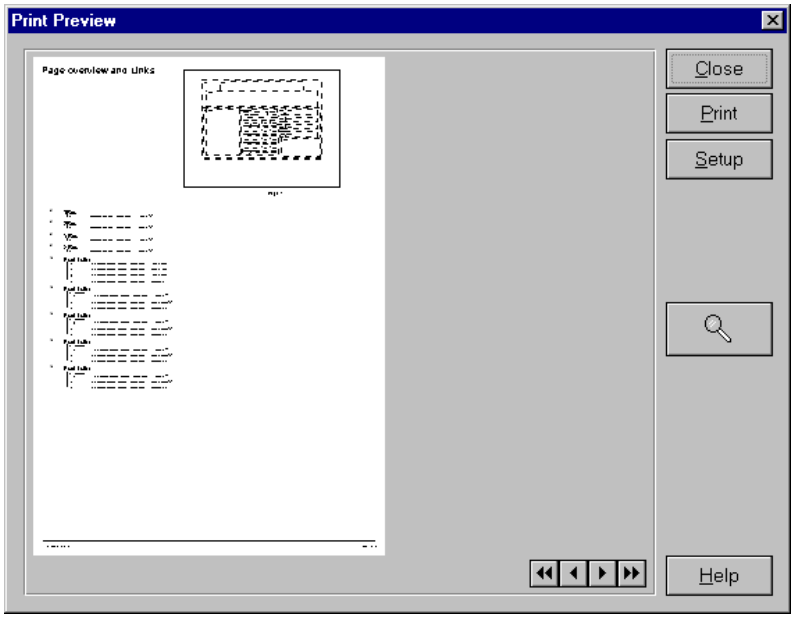
Objects and Annotations

This option opens the same dialog box as the previous ones. The printout shows a page-oriented list of the components in an

application. In addition, you will see only the annotations of each component. Again, you can choose if you want to print all pages or only specific pages.

File | Print Preview

The mindmap 'Print Preview' function allows you to look at the different layouts. It also helps you analyze your applications and create documentation, as does the **File | Print** function. The difference is that you will see a preview of the printed output on your screen. You can again choose one of the standard layouts: mindmap Page (graphical); Page Overview and Links; Object List; Objects, Links and Annotations; Objects and Annotations; or any you created yourself. In a dialog box, you can choose if you want to print only selected components, all pages or only specific pages.



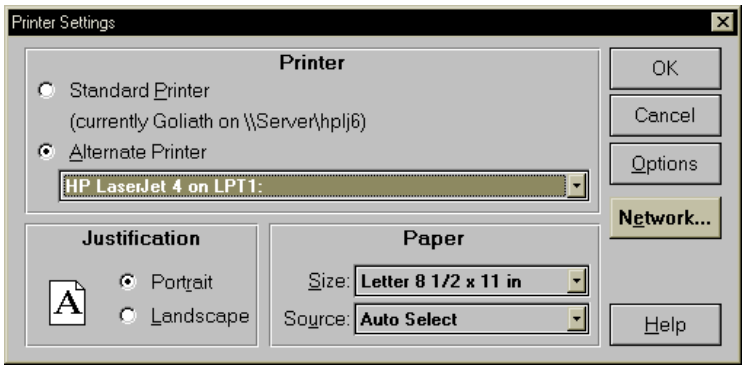
View how a *mindmap* application looks like on a printer

The Print Preview dialog box consists of a preview of the screen you have chosen or of the first of several chosen pages. In addi-

tion, you can choose one of several options: you can close the preview dialog box; you can print the page; or you can change the standard layout you have previously chosen. A button with a magnifying glass lets you zoom in and out of the preview screen. Four buttons are at the bottom of the preview dialog box. If your selection has multiple pages, you can use the arrows to move to the first page, the previous page, the next page or the last page.

File | Printer Setup

This command allows you to change the printer settings. You can choose the 'Standard Printer' or an 'Alternate Printer'. Select the appropriate orientation: 'Portrait' or 'Landscape' and the paper 'Size' or 'Source'. Depending on your installed printer, additional options may be chosen. The standard message box for printer settings of your Windows installation is being used. For more information, see your Windows manual.



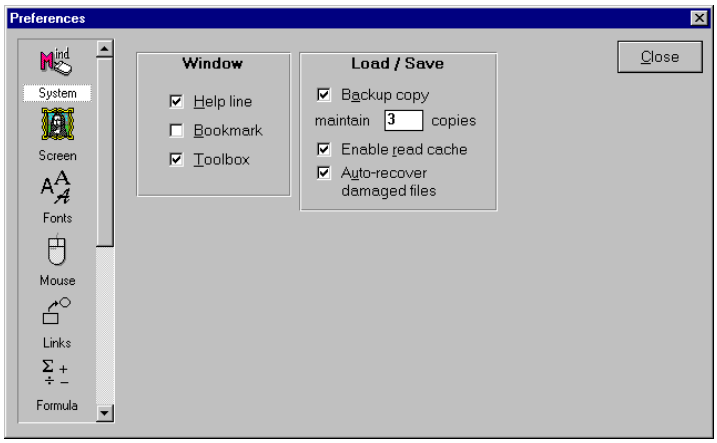
Configure the printer and additional options

File | Preferences

This menu selection offers access to numerous settings, enabling you to customize mindmap to your own specific preferences:

- System
- Screen
- Fonts
- Mouse
- Links
- Formula
- Names
- Application
- Database
- Network
- Info
- VBX

If you click on the menu command **File | Preferences**, a dialog box will appear which gives you these options to choose from. By clicking on one of the menu commands that appear in the list on the left side of the **File | Preferences** dialog box, you select which item to view.



A multipurpose dialog to configure various options

Whenever you click on one of the items in the **File | Preferences** dialog box, new options will be offered for each selection. When you click on another item, without having closed the **File | Preferences** dialog box, **mindmap** will ask you if you want to accept the changes you have made or if you want to discard the changes. This is only true if you have made previous changes to other preference settings. The dialog boxes are described as follows.

File | Preferences | System

In the System dialog box you can define whether a Help Line, a Bookmark and/or the Toolbox will be seen.

When you set the check mark of Help Line to on, the status bar at the bottom of the mindmap screen will appear. The status bar displays component names, page scroll buttons, access to the parser window, the object list, and the field in which component names can be defined. If you switch the status bar off, you can still navigate to previous page and next page by using the **F7** and **F8** function keys. Alternatively, you can use the menu commands **Page | Next Page** or **Page | Previous Page**.

If you have set the option, the name of the page where you have set a Bookmark will be shown in the right most section of the status bar.

The toolbox, which contains the most commonly used components of mindmap, appears by default on the right side of the mindmap screen. With the appropriate check mark in the System dialog box or with the shortcut function key **F10** (or **View Toolbox** in the menu **View**), you can make the toolbox visible or you can hide it.

You also have various options regarding the loading and saving of applications. In the Load and Save menu, you can define whether mindmap is to generate a backup copy of your application whenever you save it, and if mindmap is to:

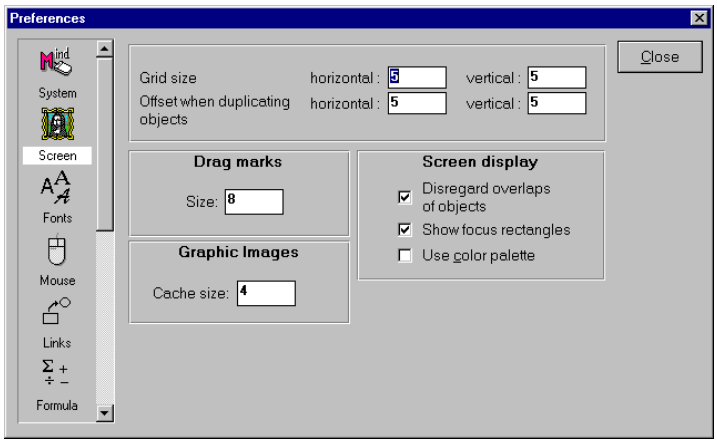
- Maintain n copies
  mindmap will distinguish backup copies by giving them the extensions .B00, .B01, etc. up to the number of backup copies you have specified. You must enter a value in the range from 3 to 50 backup copies. The default is 3.
- Enable read cache
  mindmap can enable a read cache when loading an application. Depending on the media your application is stored on, it can decrease the loading time if you enable the read cache.
- Auto-recover damaged files
  If this option is checked, mindmap will attempt to recover the contents of an application file, should it be damaged for some reason.

File | Preferences | Screen

The Grid size determines the distance between screen positions where you can place a component when moving or creating components; for example, when you specify 5 as grid size and you move a component, it 'jumps' by five pixels. The grid is invisible, yet allows you to more easily align components on the page. The size of the grid must be in the range of 1 to 100 pixels.

The Screen dialog box allows you to define an Offset when duplicating objects. The default is five pixels horizontally and vertically. This allows you to create one component of the desired size, select this component and duplicate it as often as you want. Note that some components, like buttons and input fields, have their own implementations of the duplication offset.

The size of the duplication offset must be in the range of 1 to 100 pixels.



Configure various screen oriented settings

- Drag Marks:

  These specify the size of the squares surrounding a selected component. Generally, you will not need to change this setting.

- Graphic Images:

  The setting allows you to set the number of images that are to be cached. mindmap can deal with a large amount of graphic im-

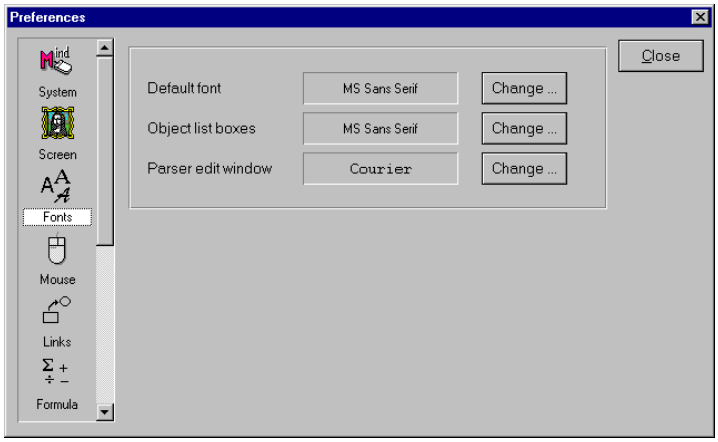
ages imported into an application. Obviously these images cannot be kept in memory simultaneously. mindmap will write images to the Windows temporary directory (usually C:\TEMP or C:\WINDOWS\TEMP) according to a 'least recently used' algorithm. This means that images that have not been visible for the longest time are written to the hard disk. The value of four images is a rule of thumb. If your application will have more than 4 different images on the same page you will want to increase this value.

- Screen Display:

  When you clear the check mark for the option Disregard overlaps of components, mindmap checks before painting any component on the screen. This is to determine whether all of it will appear or if some parts will be overlaid by other components. If parts of a component are hidden, they will not be painted. This leads to a smoother painting of the mindmap screens. Note that, depending on your video adapter, the performance of your computer can be a little bit slower with this option set to off.

File | Preferences | Fonts

You also have the option of selecting different fonts or font settings for various situations.



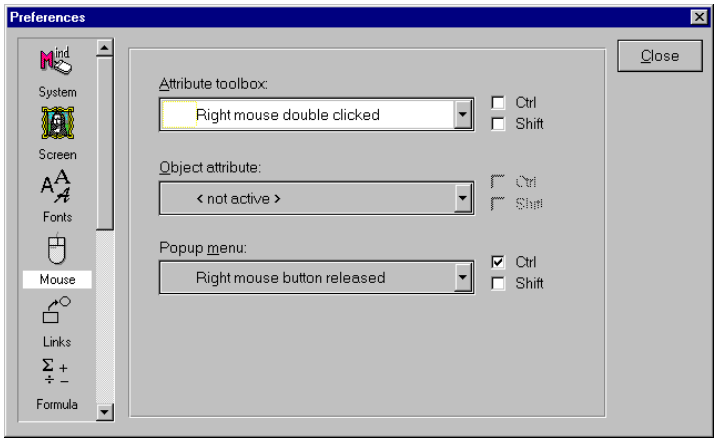
How to select default fonts

- The Default Font setting will be the standard font for all components dealing with fonts.
- The font selected for the object list will display whenever the objects are presented in list form, such as the tear-off list on the status bar, or in the link dialog boxes. Also, this font is used to display text in the status bar.
- The font setting for the parser will be reflected in any input field where parsed statements can be entered.

File | Preferences | Mouse

Throughout this manual, the term 'click' is used when an action with the mouse is being described. Depending on your settings of the File | Preferences | Mouse options, you have to choose the appropriate mouse action by yourself.

The Mouse symbol allows you to determine different kinds of mouse operations for opening the Attribute Toolbox of a component, the Object attributes or a Pop-up menu. You may configure the mouse behavior to your preference. Many Windows programs have now standardized on accessing the properties of an object/component by means of right-mouse button clicking. Therefore, this is the mindmap default setting.



Lets you specify how the buttons of your mouse react

By default, mindmap will display the attribute toolbox of a selected component by clicking once with the right mouse button (this action is called 'Right mouse button released') on the selected component.

Components that can have a focus - such as input fields, data tables, etc. - will not interpret a mouse click inside their borders as a command to display a selection. Clicking inside the border of such an object merely places the focus inside this component. If you wish to access the attributes, you must click on the border of the component. The mouse cursor will appear either as a flat hand or as a pointed index finger when it is immediately above the border.

In the attribute toolbox of a component, you may have an component specific attribute. By default, mindmap does not offer a direct method to open this attribute by clicking on a selected component. Here you can change the settings to one of the three methods 'Right mouse button released', 'Left mouse button double clicked' or 'Right mouse button double clicked'.

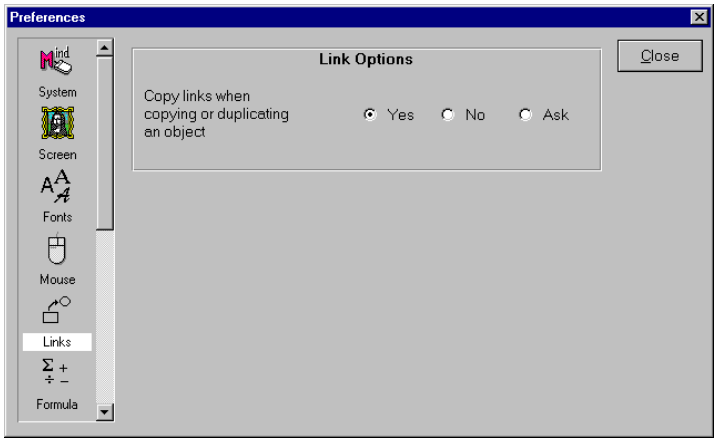
mindmap also offers the choice of opening a Pop-up menu which contains all items that are contained in the menu bar at the top of the screen. Here again, you can choose between the choices 'Right mouse button released', 'Left mouse button double clicked' or 'Right mouse button double clicked'. This feature comes in handy, when you are in full screen mode and thus don't have access to the menu which is normally at the top of the screen.

In all of the above mentioned cases, you have the option of prefacing a mouse click with either a **CTRL** or a **SHIFT** key. These modifier keys are offered so that you can distinguish between otherwise identical mouse clicks, although this does require a two-handed operation.

The preferences dialog will check your entries to make sure that no identical mouse actions are assigned to different attributes.

File | Preferences | Links

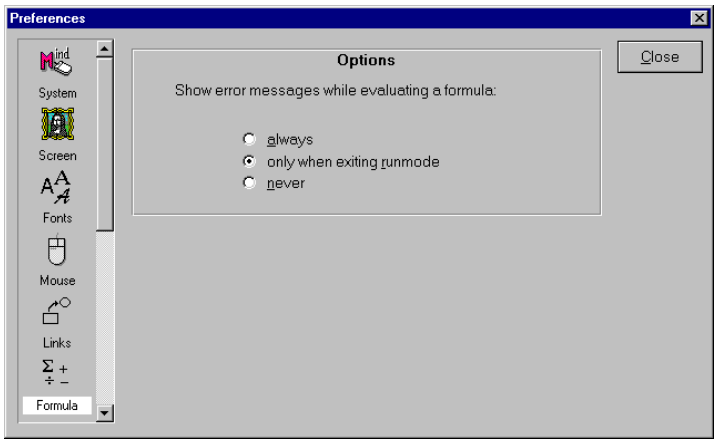
The option Links allows you to choose whether links previously placed on components are to be copied along with the component when it is copied or duplicated. When you check the option Ask, you will be asked every time you copy or duplicate a component, if you want to copy its links as well.



When you copy components, you can define how mindmap deals with the associated links in the copy process.

File | Preferences | Formula

'Formula' has the option to show error messages while evaluating a formula in three different ways: always, never or only when exiting run mode. You can view any errors that were detected by the parser as it was attempting to interpret the input.

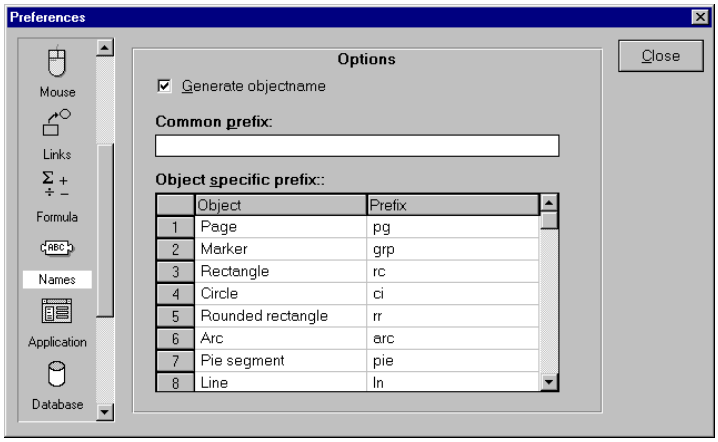


If you have an error in a formula, when would you like to be notified?

When building an application, you might wish to have messages resulting from errors in conjunction with the parser to appear immediately when encountered (Always), or never (Never), or only when terminating the application. If you choose the first option, a dialog box displaying the error message will appear during execution. You can either disregard it and deal with it later, or you can interrupt the application and attempt to fix it. If you choose the option Only when exiting run mode, a dialog box will be displayed at the moment you terminate the application. The dialog box will then display a list of all error messages encountered during the execution of the application.

File | Preferences | Names

While developing your application, it is useful to assign self-explanatory names to the different components.



Define the naming conventions for new components

The preference Names is designed to give different components a special prefix so that one can differentiate them. For example, you can define each command button to have the prefix btn. After creating your buttons, they are named btn1, btn2 and so on. To enable this feature, the check box Generate object name must be checked. You can assign a common prefix for all components by typing the desired letters and special characters into the corresponding text dialog box. This appends your prefix between the component type name and it's incremental

number. For example, if you type in the letters _name_, your buttons will be given the names btn_name_1, btn_name_2 and so on. This feature is particularly useful in team application development or application maintenance.

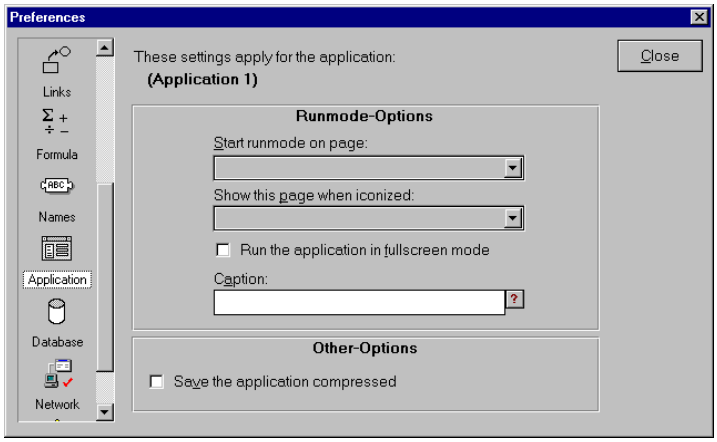
mindmap is shipped with a default set of names for all components. It is strongly recommended that you keep this default setting. In addition, it is also suggested that you always give a component a name which hints at its purpose, relationship to other components, or whatever. This makes navigating the application space much easier and it will assist you, or others, in maintaining the application.

Component	Prefix
Rectangle	rc
Rounded Rectangle	rr
Circle	ci
Pie	pie
Arc	arc
Line	ln
Push button	btn
Radio button	rad
Check box	chk
Scroll bar	scrl
Text component	tx
Imported graphic	grf
Input field	edt
List box / Combo box	lst
Data table	tbl
Database	db
MCI component	mci
Output page	rpt
Input/Output File	file
Input/Output Printer	prn
Input/Output Clipboard	clp
VBX component	Class name from VBX

mindmap requires that a naming convention be considered when assigning names to components. A component name may have a maximum of 255 characters. It should not include special characters such as blanks, question marks, etc. You should not use numerals as names. The reason for these limitations is the parser operation. If a component were to be named 1 and another one be named 2, then the parser would not know whether the statement 1+2 refers to the values of the components named 1 and 2 or to the numbers themselves. You are permitted to use any conceivable name, but mindmap warns you when you attempt to disregard the conventions.

File | Preferences | Application

When you change from edit mode to run mode, **mindmap** starts running the application, by default, at the first page of your application.



Specify how a particular application will execute

The preference setting Application allows you to specify a page that functions as the starting page in run mode: Start run mode on page:.

This facility is used if you intend to put pages at the beginning of your application which are to remain inaccessible or invisible to the user. Typically, such pages are backgrounds, locations for general purpose components, configuration pages, etc.

As an easy way to test a special function on a page that is not the first page of your application, just tell the program to start with that page. This avoids having to go through your entire application to only test one function or page. You can then easily re-set the start page to the original start page or another page you want to test.

This feature only applies to mindmap installations running on Windows 3.11 or Windows for Workgroups.

If you select one page of your application under the preference Show this page when iconized, a portion of this page is shown in Windows, when the user minimizes your application in run mode. You can use this facility to create your own desktop icons for mindmap applications. Be sure to place the bitmap at the top left hand corner of the page you have designated. Only the extreme top left 32 x 32 pixels will be shown. Make sure that the bitmap corresponds to the size requirements for desktop icons (32 x 32 pixels). You might also want to experiment with altering icons, which could be used to display various states of your application. If you do not specify a page like this, your application will show the standard mindmap icon once it has been minimized.

Choosing the option Run the application in full screen mode hides the title and menu bar. This setting is generally used when building multimedia applications, which are not to include typical Windows menus, caption bars, and/or status bars.

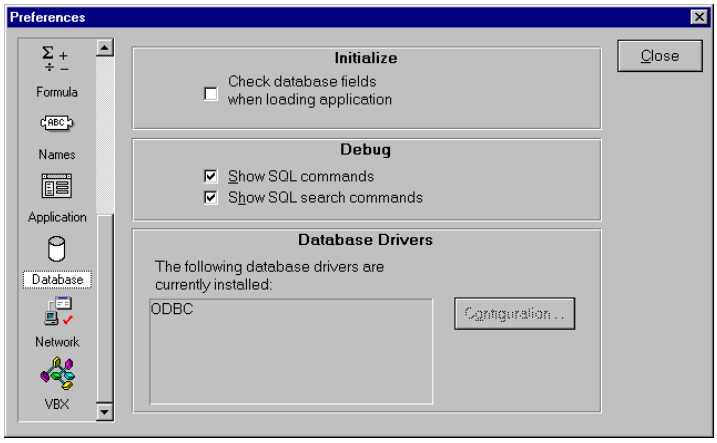
You may optionally specify a caption title which will show up in the caption bar of your application. This entry is evaluated by the parser. Therefore you may specify a component name which resolves to a string.

If you want to Save the application compressed, choose the special check box in that dialog box. mindmap then saves space on your hard disk by compressing the application. The compression algorithm mindmap employs is a compromise between minimal file size and minimal loading time. Higher compression leads to longer loading time, since the file must be decompressed. Feel free to use an external compression utility, since there definitely will be more space you can save, but you must be aware of possible performance degradation.

File | Preferences | Database

The Database preference allows you to Check database fields when loading the application during the initialization. mindmap internally saves the structure (such as the field names, types and widths) only once when a database component is placed into an application. Checking this option will cause mindmap to re-connect to the database and verify that none of the columns have changed since the application was initially created. We recommend that you clear this check box, as it will decrease the

loading time of your application. However, if you are working with an evolving database, you might want to check this option to make sure that you do not lose changes to the table structures.



Configure various database specific settings

Another option allows you to display SQL commands sent to the database driver. You can elect to Show SQL commands and/or to Show SQL search commands. This is very helpful in determining the validity of SQL commands that your application is issuing. When you dynamically construct SQL commands (via the parser), the database driver may not actually be receiving what you think it should receive. You can also see the actual SQL commands mindmap generates. Selecting one of these options will make the process more visible to you for analysis.

To see what is happening, you need to set the option View System Log in the View menu to on. You can also use the shortcut *CTRL+F10* to switch the mindmap System Log on or off. In the mindmap System Log dialog box, you can view every SQL message.

Yet another way to look at the SQL command messages is to look into the file MMERROR.LOG. mindmap writes all SQL messages into this file, provided that you have set the Logging Level in the MINDMAP.INI file to 4 or greater.

By default, mindmap assumes that the Logging Level entry is 4. If, for some reason, this parameter has been changed, please use any text editor and open the file MINDMAP.INI in the mindmap installation directory and search for this entry in the section [System]. There you will find the following statement:

```
[System]
Logginglevel=4
```

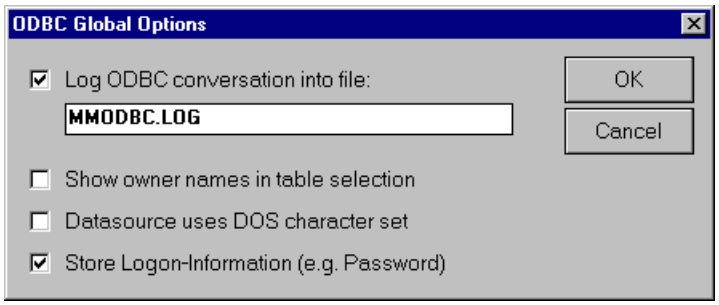
or any other value. The valid entries are:

Name	ID	Description
LOG_FATALERROR	1	Fatal errors such as 'Unrecoverable Application Errors'
LOG_ERROR	2	mindmap errors (e.g. Parser statements)
LOG_WARNING	3	mindmap warnings (e.g. Parser statements)
LOG_MESSAGE	4	Information (e.g. from the database manager)
LOG_TRACE	5	Statistics
LOG_WINDOWS	6	Internal Windows warnings

Again, make sure that this entry is set to 4 or greater.

Additionally, you will find the currently installed database modules in a list in this dialog. If supported by the particular module (as is the case for the mindmap ODBC module), you may be allowed to proceed to a more driver specific configuration. Please highlight the database driver by clicking it with the mouse and press the Configuration button.

The ODBC driver will show the following dialog:



Special configuration dialog for ODBC settings

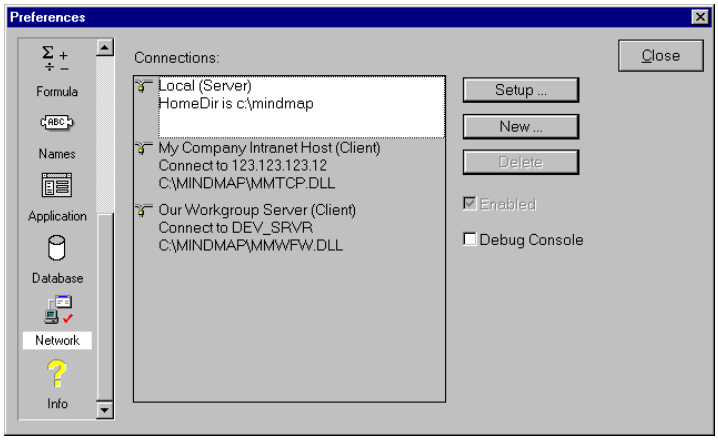
- Log ODBC conversation into file
  This dialog lets you select to append every call into the ODBC environment into the specified file. If no path is included, this file will be created in the mindmap installation directory.
- Show owner names in table selection
  Especially in a multi-user environment, different user authorizations may be allowed to use different tables having the same name. That means that duplicate entries in the table list for a particular ODBC database may appear. ODBC distinguishes between these tables by supplying a valid database user name with the table -known as the Owner of the table. By checking this option you will find the owner names as well as the table names separated by a . in every list of tables.
- Data source uses DOS character set
  If you are using a data source that has been created with the MS-DOS version of a database program, you may find language dependent characters (such as German umlaut-characters) misspelled. Check this option to let mindmap translate these special characters properly.

Checking this option means that your database passwords may be written to the MINDMAP.INI file. Generally, passwords are protected information. Be aware of the fact that others may have access to your MINDMAP.INI file and can spy out your passwords!
We strongly recommend not to use this option unless you are working with an unprotected development version of a database.
All of these settings are global to the database driver and therefore affect all database components in all of your applications.

- Store Logon Information (e.g. Password)
  If this option is checked, mindmap will save all necessary information regarding the logon procedure to a specific database system into the MINDMAP.INI file. If your database system requires an authentication, checking this option will remember your password and therefore does not require you to enter your password again upon each new connection to the database.

File | Preferences | Network

This option allows you to define the various network connections to the servers. The servers here are meant in the physical sense, that is, the computers to which a connection is to be established.



Overview of existing Client/ Server connections

➤ A detailed discussion of these settings can be found in the chapter dealing with client/server technology which starts on page 317.

File | Preferences | Info

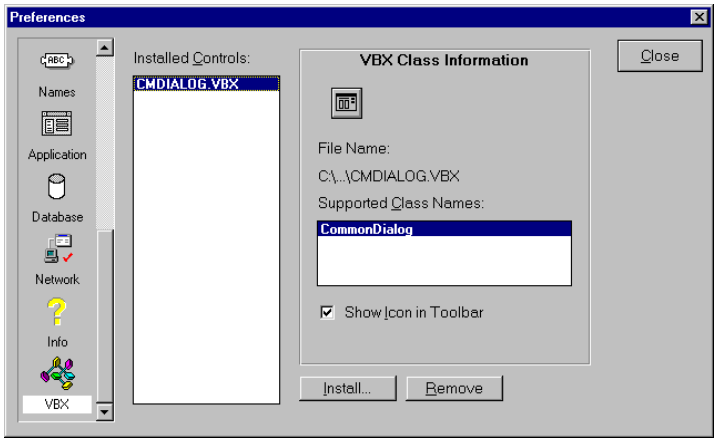
In the preference menu, general information is available. Info gives you information about the libraries for the different mindmap components.

You can view an Object list. Clicking this option outputs a list of all components of the current application to an external text file. You are prompted to key in the name of this text file and will then be asked if you want to review it immediately. If you indicate Yes, mindmap automatically loads the file in a standard Windows word processor, such as 'Notepad.'

The Global Memory, the Error Log and the System Heap can be viewed and output. This can be used for support purposes, such as when you use our support system. The contents of the message boxes that will be generated, can be forwarded to the mindmap support staff.

File | Preferences | VBX

The VBX preference allows you to Install or Remove Visual Basic controls. This is a powerful facility that gives you access to a huge library of Visual Basic controls or eXtensions (VBX). You can quickly and easily integrate new features into mindmap with this option.



Install and Uninstall external VBX controls

To install a VBX component, click on Install and search for the desired file with the extension *.VBX. Most of the VBX files can be found in the WINDOWS\SYSTEM directory. After choosing the VBX component, its name is shown and the Supported Class Names are listed. mindmap must now be restarted, so that all modifications will take effect. After the restart, an icon in

the toolbox will symbolize the VBX component and you can work with it, as with any other mindmap component. The VBX component is also listed in the Objects menu of the menu bar.

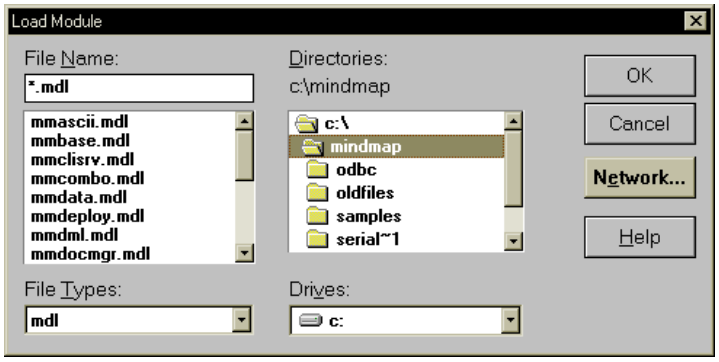
You can remove a VBX control as easily as you can install it. In the Installed controls list box, highlight the name of the VBX component that you want to remove in the Installed controls list box and click on the Remove button. mindmap then asks you if you are sure that you want to remove this control. If you click on the Yes button, the control will be removed. Note that this modification also becomes effective only after you have re-started mindmap. Note that removing the VBX from mindmap does not delete it from your system.

VBX implementation notes: There are certain limitations to which VBXs can be installed in mindmap. Any VBX requiring facilities built into Visual Basic can obviously not be used, since Visual Basic is not installed or running inside of mindmap. Currently, Microsoft only allows third party environments (such as mindmap) to support VBXs which comply with the VBX 1.0 standard. Thus, only those VBXs which support at least VBX 1.0 specifications can be installed in mindmap. You must determine the compatibility by either reading the documentation supplied with the VBX or by contacting the vendor. Also note that mindmap has no method of knowing or modifying any behavior of a VBX, other than what the original developer of the specific VBX has allowed. For example, if they provide the capability for you to change a color in the VBX, then mindmap will allow you to do so. Also, when you deploy your application that incorporates the functionality of the VBX, the VBX must be deployed with the application, which means you must have the necessary license to do so. This is beyond the purview of mindmap and must be dealt with between the individual developer and the owner of the VBX.

➤ For more information about VBX controls please refer to the chapter about VBX components on page 226.

File | Load Module

mindmap is a module oriented program. When you start mindmap, numerous mindmap modules are loaded. (You can observe the process in the status bar.)



This dialog lets you load additional *mindmap* modules

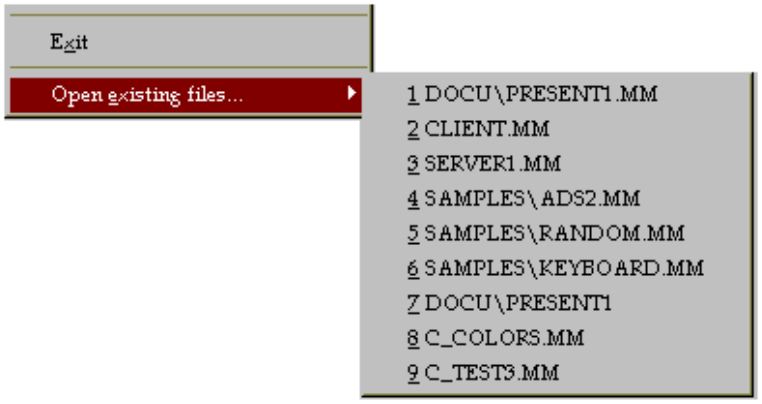
Whenever you want to load additional modules, choose the option Load Module in the File menu. You then select a file with a *.MDL extension that loads the appropriate module. This facility is only necessary when you wish to load a mindmap module (MDL file containing components), after mindmap has been loaded initially. In general, mindmap installs all modules (MDL files) it finds in its default directory and those specifically pointed to in the MINDMAP.INI file.

File | Exit

If you want to leave mindmap, choose Exit in the file menu. mindmap will prompt you to save your application if you either made, or attempted to make, any changes while the file was open.

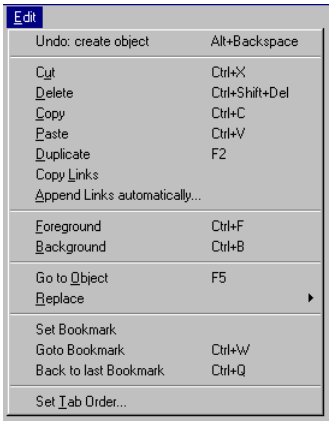
File | Open existing files

This command is only available if you have previously saved any mindmap files. The most recently saved file stays on top of the list. Up to nine files are listed and can be chosen very quickly. A More option shows additional older files.



Quickly load *mindmap* applications with which you have recently worked

Edit Menu

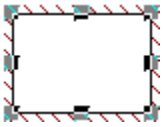


Edit Menu

Since editing is one of the most common tasks in application development, we will offer some general information about the selecting, sizing and dragging of components. After this, we will describe the items that you can find in the Edit Menu.

Selecting Components

A number of actions dealing with components can only be performed if the component has been selected. Clicking with the pointer on a component selects it. The component will receive a selector indication, i.e., a hatched border with rectangles (see also **File | Preferences | Screen** on page 42) in the corners and along the borders of the component.



Selected rectangle with hatched border displayed

In some cases, you can click anywhere on the component in order to select it. If the component is a type which accepts a focus in run mode (input fields, radio buttons, etc.), you must select the component by clicking on its border. The color used to draw that border can be set by the MarkerColor keyword in the [System] section of MINDMAP.INI. It uses RGB format and is red (255,0,0) by default.

You can also select components in other ways:

1. Hold the left mouse button down and move the mouse cursor around one or more components by starting somewhere outside the lower right of the component or group of components and moving somewhere to the upper left of the component or group of components. You don't need to move the mouse cursor totally around the component or group of components -- just 'touching' them is enough to select them. Once you have finished selecting the desired components,

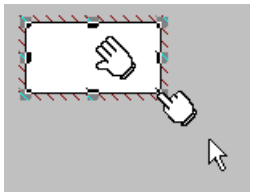
release the mouse button. You will then see a selection marker around the entire group. This is the fastest method to select a group of components.

2. The second method is to move the mouse cursor from outside of the upper left of the component or group of components to the lower right of the component or group of components. Once you have finished selecting the desired components, release the mouse button. You will then see a selection marker around the entire group. In this case, only components that have been enclosed completely by the mouse cursor are selected.
3. Another way to select several non-contiguous components is to hold down the **SHIFT**-key and select the desired components by clicking on them with the mouse cursor.
4. In addition, you can select a component by using the Object List. Expand the list, navigate the list until the name of the component to be selected is visible. Now press the **SHIFT** key, keep it pressed and click on the component with the mouse. See the explanation in the section **Edit | Go to Object** on page 70 and especially the note in Component List. This is especially useful in locating small or otherwise hidden components in your application.

If you select components using the methods described in 1) or 2) above, then a temporary group will be created. If you now attempt to access the attributes, you will notice that the set of available attributes might have changed, depending on the individual components contained in the group.

Sizing Components

The selector rectangles around a component have different functions. If you move the pointer into one of the drag marks, the cursor will change from an arrow to a hand with a pointing index finger. This means that you can now enlarge or shrink the component by pressing the left mouse-button and dragging the component to the desired size.



Component with different drag points

If you drag the component by a rectangle in the middle of the border, the component changes its size horizontally or vertically. If you drag the component by a rectangle in the corner of the border, you will change its size: either horizontally or vertically or both. In this case, when you also press and hold the **SHIFT** key, the component will keep it's width/height ratio.

By using the corner rectangle, a component can get a rough shape and with the middle rectangles, one can precisely define the height and width of a component.

Various fields in the status bar

The status bar can be very helpful in determining the exact size and location of a component. The status bar states the class (component type) and the name of a selected component. It also displays the position of the component in rounded brackets and the size in square brackets (horizontal and vertical position in pixels). The position is defined by the upper left pixel of the component.

➤ See also the parser functions xPos and yPos on page 450.

Dragging Components

Whenever the pointer moves into a selected component, the cursor changes to a flat hand. (If the component you are dealing with can receive a focus, then the cursor will only appear as a flat hand, when it is immediately above the border of the component.) The flat hand symbolizes that you can drag the com-

ponent by pressing the left mouse button. Releasing the mouse button will leave the component in its current position.

Edit | Undo

mindmap is capable of undoing most actions relating to the drawing of components. Any operations to the file system, to the printer, or in conjunction with any other external devices, cannot be undone. Since the history of all actions must be stored in local memory, there are also some limitations as to how far back you can undo operations. You also cannot undo operations relating to attributes and links.

After you start mindmap, the command Undo not available will appear in the Edit menu. From then on, most actions will be recorded and can be undone. If not specified otherwise, mindmap will save the last 40 actions in its local undo buffer.

If you group components together and move or resize them, mindmap will save this action as if every single component were moved or resized separately, therefore occupying multiple steps in the undo buffer.

Edit | Cut

This is a common Windows command. It is used by mindmap in the same manner as in most other programs. You can cut and paste components between different Windows programs, within a mindmap application or between different mindmap applications. If you are not familiar with this command, please remember the shortcut key **CTRL+X** for cutting, as well as **CTRL+V** as the shortcut key for inserting. The cut command copies selected components into the Windows clipboard, where it can subsequently be used for different purposes. It also deletes the original component from mindmap.

➤ See also **Edit | Copy** on page 64.

mindmap places the components onto the clipboard in various formats.

Edit | Delete

This command deletes selected components. If you have been too quick with this command, don't worry - the Undo command will help. Another possibility to undo your actions is to exit a mindmap application at this point without saving. For example, imagine you are developing an application and you have saved it as a file with the name DEMO.MM. You then proceed to make alterations to your application only to realize, that they weren't such a great idea after all. If you exit the application without saving it, you can then revert to the previously stored version to regain your initial position.

If you have links on the component (or components) that you are trying to delete, mindmap prompts you to confirm that the components have links to other components that will be permanently deleted. The **Edit | Undo** command will not be available to restore the links, once deleted.

Edit | Copy

This command copies a selected component into the clipboard. Whenever you use the **Edit | Paste** function, the component that has been copied will be placed at the same screen position it was copied from. The component is stored in the clipboard as long as no other component replaces it. This command can be used to place identical components at the same position on the same or different pages (e.g. titles or navigation buttons), or in different mindmap applications. If you paste a component on the same page as it was copied from, be aware of the fact that the copied component lies exactly over its original, making no visible difference. Move the pasted component using the pointer to verify that the paste operation has been properly completed.

In addition, you can not only copy components, but also the values of components. If you highlight the displayed value of an input field that shows a text, you can copy that text to another input field or another component, by cutting and pasting it or by copying and pasting it. Please note also that the Windows clipboard acts in different ways if you copy components and links. Most components are copied in mindmap format (for past-

ing the copied component to another position in mindmap) and in Windows MetaFile Format, enabling you to paste the component into other applications. Links are copied in mindmap format (for pasting the copied link to another position in mindmap) and as a bitmap file for pasting the link to other applications.

Edit | Paste

This command copies the content of the clipboard into the current page of the active application. For more details, see the **Edit | Copy** function or the **Edit | Cut** function.

Edit | Duplicate

This command copies selected components and places them on the screen, with a predefined offset. Command buttons, check boxes and radio buttons are duplicated vertically. That is, they are located in a column, so that they are placed correctly. The spacing can be changed in the **File | Preference | Screen** menu. The other components are duplicated with a spacing that is fixed. You can change this spacing with the function Offset when duplicating objects in the **File | Preference | Screen**. The difference between Cut/ Paste and Duplicate is that the former copies a component at the same position (via the clipboard) and the latter copies a component in a new location (slightly off-set). You can also duplicate a selected component by using the **F2** key.

Note that some components react differently to duplication operations (**F2**):

- Command buttons: The new push-button is created below the original with the vertical grid displacement in between.
- Check boxes and radio buttons (normal style): The new component is created immediately below the original with no horizontal displacement.
- Check boxes and radio buttons (push-button style): The new component is created to the right of the original with no displacement.

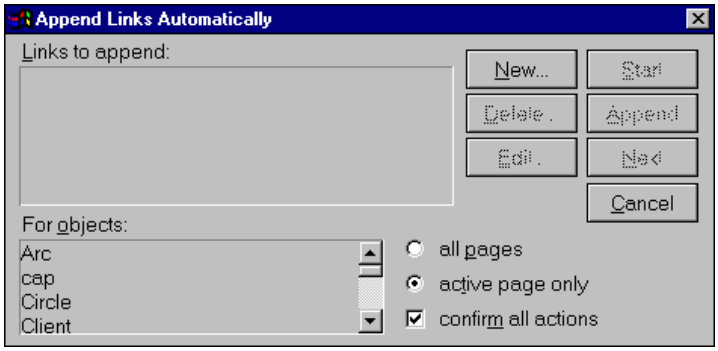
- Input fields: A duplicated input field appears below its original with no displacement, but note that a small displacement appears because child windows are created in an inflated rectangle (by a size of 4x4 pixels).

Edit | Copy Links

This command is useful when copying recurring links. An example would be copying cursor changes as a result of mouse movements into or out of components. To copy links, select a component that contains the corresponding links and copy the links by clicking on that command. Now select the component that is to receive the links and open the Link function in the Properties menu (shortcut key **F6**). Click on **Edit | Paste** and the links will be pasted onto the selected component.

Edit | Append links automatically

When developing a mindmap application, you will sometimes desire to use the same links in multiple locations. Before placing or copying all these links individually, use the convenient command Append links automatically. A dialog box will appear. In a list, you select the component type that will get the links, e.g., a command button. Then click on New and select the appropriate link, e.g., 'when mouse moves into component change cursor'. Before you assign the links, you must choose if this command applies to the components of all pages or to those of the active page only. The Start button executes the command immediately.



Create links to be appended automatically

If you have set a check mark to confirm all actions and you click the Start button, mindmap jumps to the first component. Now you click on Append to append the link to this component or you can click on the Next button, such that mindmap jumps to the next component without appending the link. In this manner, you may be selective about which components receive the links.

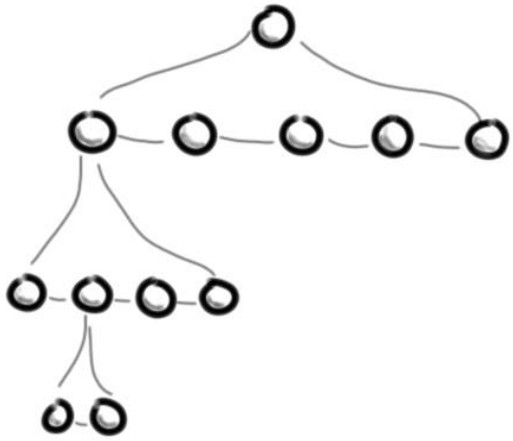
Edit | Foreground

If you have more than one component on a page, some of the components may overlap. mindmap places the components in the foreground, depending on their initial order of creation. In other words, those components you create first on a page are placed in background, the next are layered on top, and so on. The last component placed is on the top layer -- it is in the foreground.

You can change the sequence of the components, and the order in which mindmap will evaluate them in run mode, by assigning the components to foreground or to background. If you move a button onto a picture, the button will disappear if it was created before the picture. If you select the picture and place it in the background, the button will then appear. The order in which you have placed components and the subsequent rearrangement of components in fore- and background respectively, determines in which order mindmap paints the screen and in which order you can tab to components that receive focus.

mindmap works from background to foreground or from first created components to later created components. In other words, the order in which mindmap looks through each component, if there is a link to be evaluated, is from background to foreground.

The concept of foreground and background is more easily understood by looking at the following graphical representation.



The necklace of necklaces, illustrating the organization of pages, and components

The top pearl in this chain represents the application itself. The next level, containing 5 pearls, represent the pages of the application. The third level -- with four pearls -- symbolizes four components on the first page. The second component from the left, is a group component, which consists of the two components displayed immediately beneath it. When the user initiates an event (a mouse click for example), the application is the first entity to receive it. The application then forwards it to the first page. The first page then determines if components have been placed on the page. If this is the case, then the event is forwarded further down the chain into the first component. If the message is not intended for this component, it is passed on to the next pearl in the chain. In this example, the next pearl is a placeholder, a group component. It takes the event and passes it down to the first member of the group. If both of these

How event passing occurs, depends on the particular event. In general, all mouse related events are passed from *right to left*, all informational events are passed from *left to right*. Remember that the leftmost components are those in background. The rightmost components are those in foreground. This means that an event such as Left mouse button released is sent to the foreground component first, thus letting the foremost of two overlapping components receive this event (as one would expect). An event such as Application started is passed to the leftmost component first, meaning that the components will receive this event from background to foreground.

don't know how to deal with the event, it is forwarded back up to the component level and passed to the right.

Moving a component from foreground to background is actually rearranging the sequence of the pearls on the chain. Creating a group is really inserting a new component (which does not have a visual representation) and dropping the affected components down one level.

With this in mind, you can control how the application evaluates the components of a page looking for links, page after page.

➤ See also **Edit | Set Tab Order** on page 75.

Edit | Background

The Edit | Background and Edit | Foreground commands change the order of the components. Selecting a component and then assigning it to background, makes it the first component to be painted and the first to be processed, assuming links are associated with it. Conversely, moving a component to foreground will make it be painted and processed last.

Placing a component in background

In the first diagram, component number 1 is in foreground. If selected and placed in background (**Edit | Background**), it will be 'behind' component Number 3, as shown in the second diagram. It thus will be painted first and also processed first.

Edit | Go to Object

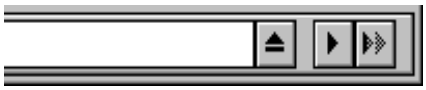
The shortcut function key **F5** or the Edit | Go to Object command allows you to jump to a specific component. You can jump to any component on any page of your application (or to any other mindmap application which is open at the time) and select it.

This is another reason why you should use self-explanatory names when naming your components. mindmap names the components automatically by setting a counter after the name prefix, but if you name the important or key components individually, application development is easier because you will recognize them faster.

Pressing the shortcut function key **F5** opens a list with all the components in your program. Now you can choose any component to which you want to go. The list shows all components of all applications which are open, but you can set a filter for specific component types, such as for a definite command button. Select the appropriate component in the list Object type.

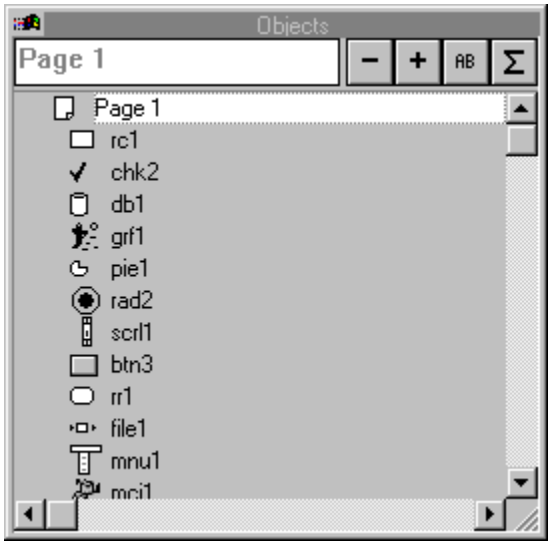
Component List

Another way to quickly jump to any of the components in the currently active application is to use the component list at the lower right corner of the mindmap screen.



Access the component list by clicking the button next to the component name editor

Just to the right of the text field which shows the name of the page (or of any selected component) you will see an arrow pointing up. Click on this arrow and the list with all components will appear.



Status bar showing the object list

You now have several options:

- clicking on the - sign shows only page components
- clicking on the + sign shows components of all types, ordered by pages (the default view)
- clicking on the AB sign sorts the component list alphabetically. Now you can enter any letter, or combination of letters, in the white text field to the left of the - sign and the component list will show components whose names start with the letters you have keyed in.
- clicking on the little collection of rectangles will display the components in the order of creation by pages.

When you find the name of the component you are looking for in the component list, press the **SHIFT** key and click on the name of the component. mindmap will jump to the page where the component is placed and will select the component. Instead of pressing the **SHIFT** key and clicking on the name of the component, you can also press the **ENTER** key while the **SHIFT** key is pressed.

If you drag the components list a little bit away from its original position, it will stay on top of the mindmap screen until you close mindmap or until you close the components list. This allows easier navigation within your application.

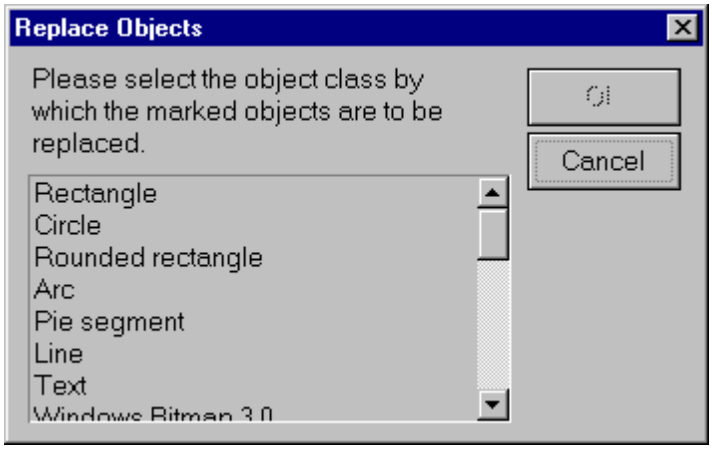
Edit | Replace

Sometimes it becomes desirable to change either components or attributes of components on a global basis. In other words, instead of editing each individual instance of a component, you can make a replacement for all instances of a component. There are two options:

- you can either replace components or
- attributes of components.

Edit | Replace | Object

Sometimes, you may find that you need to change components. Let us say you created a text component and you decide afterwards to change it into an input field. The command Edit | Replace | Object is used to change types of components.

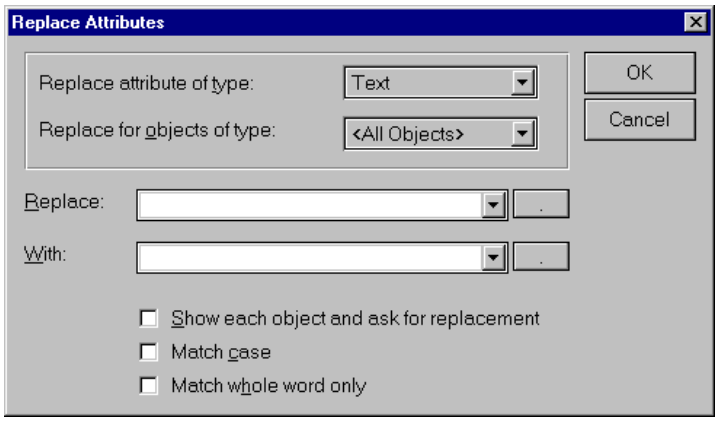


Select the type of component with which you want the currently selected component to be replaced

Select the components to be replaced and execute Edit | Replace | Object. Then select the component class (this means the type of components) which will replace the selected components.

Edit | Replace | Attribute

Sometimes you need to change the attributes of a range of components. Suppose you want to change the font of text fields. You can use the function Edit | Replace | Attribute to change one attribute type to another.



Select which type of attribute you want to have changed

In this dialog box, you can select the type of the attribute (in this case . 'Text') and for which type of components you want to perform the replacement (e.g., All Objects or Push Buttons only). From the two list boxes in the dialog box, you can select the replacement.

Begin by dropping down the first list box which refers to the attribute. Select the attribute you wish to replace.

Attribute type	Description
Text	Replaces the given text in all text oriented components such as the text component and input fields (if they are set to contain strings).
Color	Replaces the specified color to another one, if the particular object deals with colors.
Line	Replaces the specified line style to another one, if the particular object deals with line styles.

Effect	Replaces the specified effect to another one, if the particular object deals with effects.
Font	Replaces the specified font to another one, if the particular object deals with fonts.
Formula	Replaces the given text in all places that deal with parser statements including all links and link conditions. This option is useful, should you decide to rename a component which has been used in multiple parser statements.

Next determine which component(s) for which you wish the replacement to be valid. You can apply various filters for the selection of components. The next step is to specify the actual attribute value which is to be replaced. Finally, state the value it is to be replaced with. An example would be:

'Replace the Font attribute of all selected instances of command buttons from Helvetica 14 pt. to Times Roman 14 pt.'

You can also choose to Show each object and ask for replacement, to Match a case or to Match a whole word only. This facility allows you to step through the list of components and decide on a case by case basis.

Edit | Set Bookmark

While developing your application, you might want to jump between different pages that are not immediately adjacent. You can set a bookmark on the page you wish to jump to often. This bookmark can be changed as often as you wish.

For example, let's say you set a bookmark on page 5 and you are on page 23. Execute the command **Edit | Go To Bookmark** and mindmap will jump to page 5. Back to last Bookmark brings you back to page 23 again. The **File | Preferences | System** function offers the option to show the bookmark. If you have set this option, the actual page name with the bookmark is shown in the right corner of the status bar.

Edit | Go To Bookmark

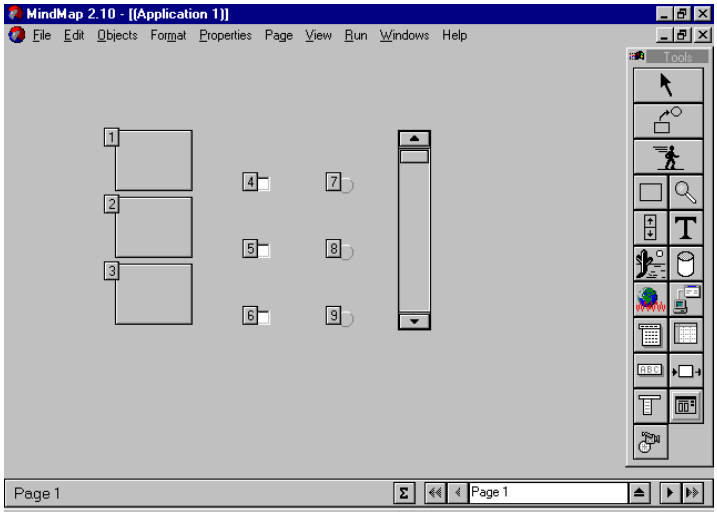
Before you can go to a bookmark, you must set it. This can be done on any page. You can quickly jump to this bookmark with the shortcut key **CTRL+W**.

Edit | Back to last Bookmark

mindmap remembers the page from where you jumped to a bookmark and automatically saves it as last Bookmark. You can jump from one page to another by choosing either Edit | Go To Bookmark or Edit | Back to last Bookmark. You can also use the keyboard shortcuts **CTRL+W** and **CTRL+Q** for this purpose.

Edit | Set Tab Order

Many applications require screens that have fields (as well as buttons) which are to be used in a defined sequence. The sequence specifies in which order each field or button is to receive the focus. The focus can be set by having the user either press the **TAB** or the **ENTER** key. The order of processing of components on the page is identical to the order of painting them. This order is initially determined by the order of their creation. If you wish to change this order to reflect some other sequence, you can either move components from foreground to background, or you can interactively change their tab order. To do this, select the menu option Edit | Set Tab Order.



This is how your application looks if the option Set Tab Order is activated

You can define a selected component to be the first in the tab order by setting it to Background with the help of the Edit | Background function.

For example, if you have a screen with some input fields, you can determine the order of their focus. With the Tab key you can allow the user to switch from one input field to the next. To do this, select the command Edit | Set Tab Order. You will see small buttons at the upper left corner of each component that can have focus. These buttons are numbered according to the tab order of the component. If you want to change this order, click on the components in the order you desire. The button number changes to reflect the new order. When you are finished renumbering, click on Set Tab Order again (or press the **ESC** key) and the small numbered buttons in the upper left corner of the components will disappear.

Please be aware of the fact that only components that can deal with the concept of the input focus will be able to display a numbered button.

Objects Menu

The objects menu allows you to choose any of the available components and place them on the screen. This method is an alternative to the selection of components from the toolbox.

Since the toolbox is the easiest method of selecting the most often used components, a detailed description of those components will be offered in the section relating to the toolbox. There we will also describe the attributes of each component and the links they can have to other components. Here we will only describe actions that cannot be done with the toolbox, but that must be activated exclusively in the objects menu.

Let's begin by describing how to place components on the screen. To place any component on your screen - for example a rectangle - choose the desired type of component from the toolbox or from the Objects menu. The appearance of the cursor will change to reflect the selected component. It generally will be displayed as a crosshair with an attached symbol. Next, set the mouse cursor to the place on the screen where you want the upper left corner of the component to appear. Then press and hold the left mouse button, pull the rectangle open as it displays the current size and release the mouse button. The component is created and placed on the screen. If you are not satisfied with the size or location of the component, you can resize or move it as described in the sections Selecting Objects, Sizing Objects or Dragging Objects.

Some components do not register an icon on the toolbox. A component is always required to register an entry in the Objects menu. Therefore, you will see more components in the Objects menu than are available on the toolbox.

Next, let's look at how to create a new page: Since mindmap is page oriented, you can only place new components on an existing page. You cannot create a new page by using the toolbox or the Objects menu. All actions dealing with pages must be issued by using one of the functions in the page menu, described later in this chapter, or by using the page buttons on the status bar.

To create a new page using the status bar method, simply click on the single right hand arrow next to the component name field. This will either move you to the next adjacent page in your application or, if one does not exist, it will warn you and allow you to have one created, if you select Yes. Selecting No simply leaves you on the last page without creating another.

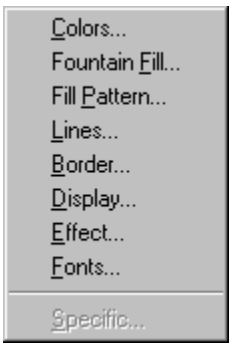
mindmap offers the following components for your use:

- Rectangle
- Rounded Rectangle
- Circle
- Pie
- Arc
- Line
- Push button
- Radio button
- Check box
- Scroll bar

- Text
- Printer component
- Database Manager
- Encapsulator (Client / Server)
- List box
- Combo box
- Data table
- Input field
- Input / Output
- Menu
- Output page
- VBX (if any VBX control has been installed)
- MCI

➤ See section Chapter 5 Component Types on page 117 for more information about the components.

Format Menu

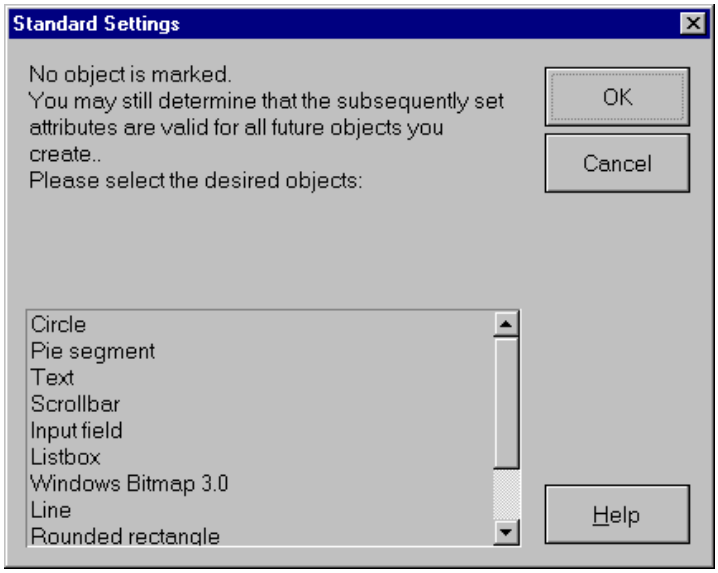


Format Menu

This menu allows you to change the attributes of components. Depending on whether or not a component is selected, the menu selection will either affect this specific instance of the component or all instances of subsequently created components. In other words, you can use this menu command to set default values for all subsequently placed components in this particular application.

First, you can select one or more components and then make the setting changes in one or more of the options in the Format menu. Once you have selected one of the attributes, the appropriate dialog box displaying the available settings for the attribute will appear.

Second, you can choose one option out of the Format menu without having selected a component. Then mindmap will remind you in a message box that there is no component selected.



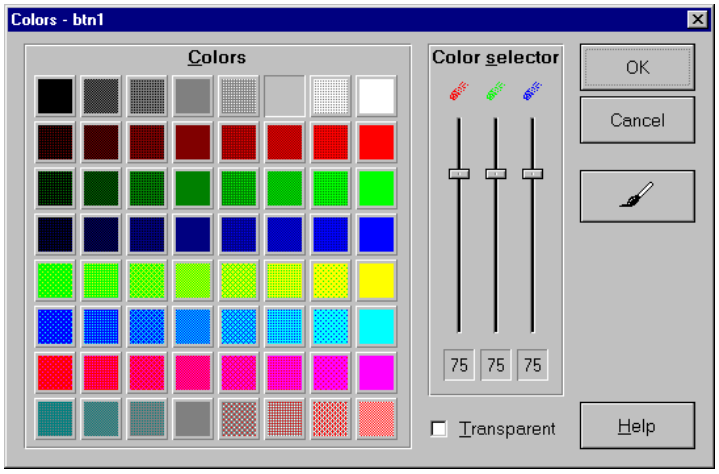
Dialog box to select default setting for component

You may then choose if the subsequently set attributes are valid for all future instances of the selected type of component you create, by choosing the desired components out of a list in the message box.

Format | Colors

You can also set the color of a bitmap to be transparent. In this case, you must also define a color of the bitmap which you would like to become transparent. This allows you to produce an effect called chroma-keying. Please refer to the section dealing with Graphics for further information.

This command allows you to change the color of components. Selecting this command opens a dialog box where you can choose a desired color, by clicking on the appropriate colored rectangle. The Color selector shows the number of the selected color as represented by the values of the three colors red, green and blue. Using the three sliders, you may modify or create a completely new color.



Select your favorite color for a component

If you wish a component to be transparent, check the option on the lower part of the dialog box. Thus, when you place a transparent component on top of another component, you can see the underlying component. This is quite useful in creating invisible hot spots on graphics, for example.

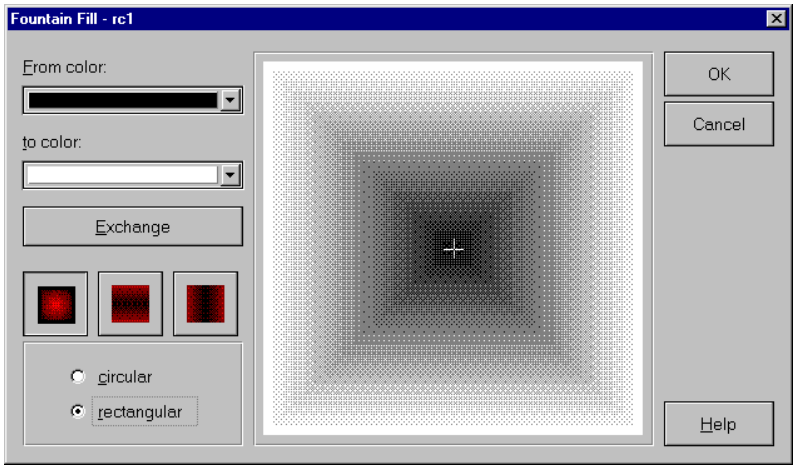
In order to change the color of an arrow head of the line component, you must choose the Fill Color option, as described here. Setting the Line Color of a line merely sets the color of the line itself and not the arrowhead.

Below the OK button and the Cancel button in the color pop-up dialog box, you can see a brush. (If the selected component is capable of having a border, then the border icon will be immediately beneath the OK and Cancel button.) Clicking on that button allows you to define the color of the border of a component. This option can also be reached by using Format Lines. (If the selected component is capable of dealing with the Fountain Fill and/ or Fill Pattern, then these will also appear on the Attribute Toolbox.)

Depending on how the color dialog has been opened, there are two other options underneath the brush. The first is Fountain fill to give draw components special effects like color toning. This option can also be reached via **Format | Fountain Fill**. The second option is Fill Pattern. This option can also be reached via **Format | Fill Pattern**.

Format | Fountain Fill

Fountain Fill assigns a gradient color fill to the selected component(s).



Define a fountain fill

Choose the first color out of the menu From color and define the second color as the To color. Now you can see a gradient fill

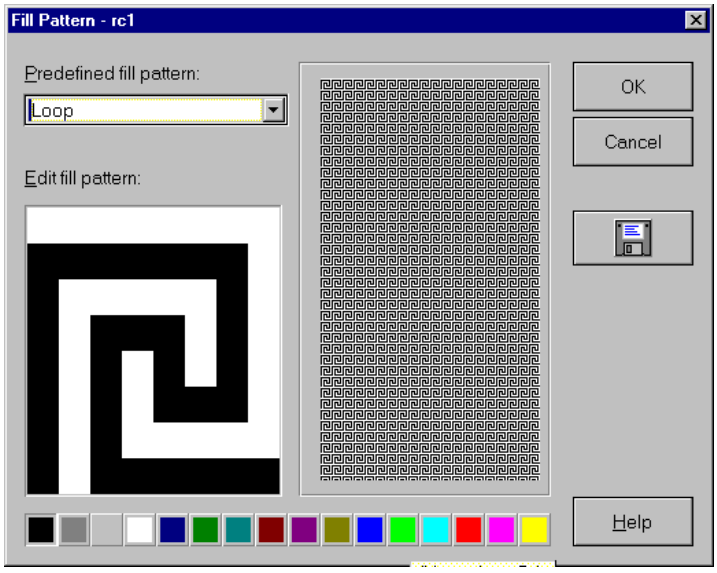
ranging from the first color to the second color and which can be modified in different ways:

- with Exchange, you can reverse the colors;
- with the three command buttons, you can set the gradient fill to center, horizontal or vertical;
- and you can give the gradient fill a rectangular or a circular shape.

In the preview box, which displays the result of the current settings, you can see a little cross-hair. You can grab this cross-hair with the mouse and drag it around by keeping the Left mouse button pressed. In doing so, you can determine the focal point of the gradient fill.

Format | Fill Pattern

Components can be assigned special patterns, such that they have an distinctive appearance.



Define or load various patterns

The fill pattern option only permits the usage of the 16 standard colors for editing. If you load an external bitmap as a pattern and intend to modify it, it will be mapped down to these colors.

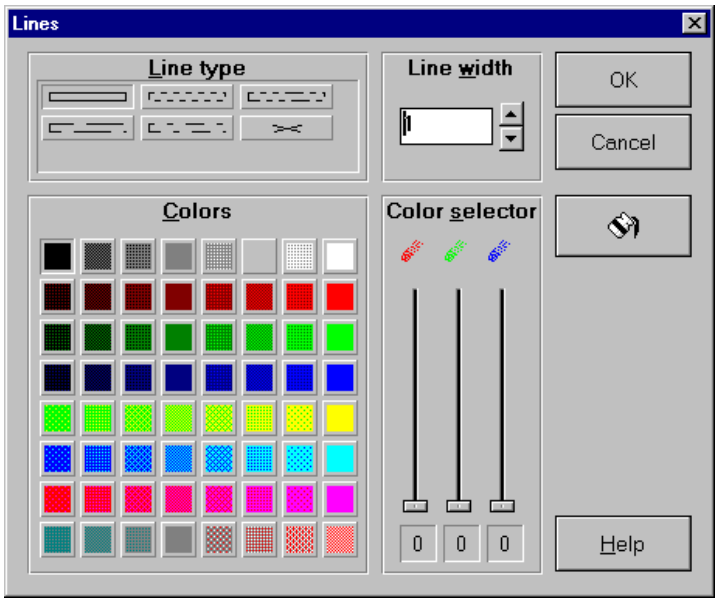
A selection of patterns is predefined, such as a loop, vertical double-lines, a small box, etc. The preview box on the right side displays the current setting. Just to the left of the display, you can view an enlarged version of the currently selected pattern. You can use this display in a manner similar to a bitmap editor. Pick up a color from one of the buttons below the editor box and begin to place the (enlarged) bits in the pattern. Immediately, you can see the effects on the pattern in the box on the right.

This dialog box also offers the option of loading a bitmap from the file system. Click on the button displaying a diskette and select the desired file. Once loaded, it will be visible in the editor. Again, you can choose to manipulate the bits, if you wish.

Format | Lines

If you select a line style other than solid, only a line width of 1 is supported.

This menu allows you to define the appearance of the border of a component. If the selected component happens to be a line, then the settings are interpreted by the line itself.



Set the line attribute for a component

Not all colors may be accepted by the line depending on the settings of your graphic adapter. In general, only video adapters that support more than 256 colors allow the selection of the full set of available colors. Experimentation may be required for a color suitable for your application.

The upper left part of the dialog box offers various settings regarding the structure of the line. Multiple options for dashed and solid structures are offered. If you wish to have a transparent line or border, select the button displaying an 'x'. Again, this is useful for creating an invisible component on the screen.

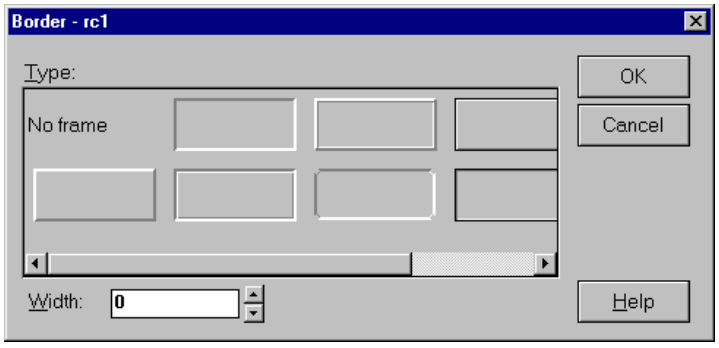
On the upper right part of the dialog box, you can set the line width. The minimum line width is 1 and the maximum width is 100.

The lower part of the dialog box allows you to set the desired color. You can either select one of the buttons displaying a color, or you can use the sliders to mix your own color.

Format | Border

You can define special borders for three component classes:

- text components
- input / output and
- rectangles.

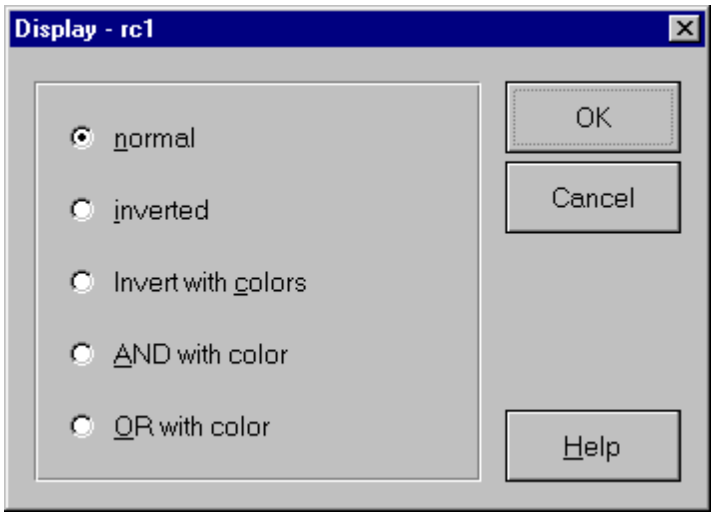


Change the appearance of a component by selecting from a list of predefined borders

Select one of the seven predefined borders. By altering the Width of the border, you can further influence the appearance.

Format | Display

With this command, text components, rectangles, rounded rectangles, and circles can be formatted with certain attributes like inverted, invert with colors, and with color, as well as or with color.



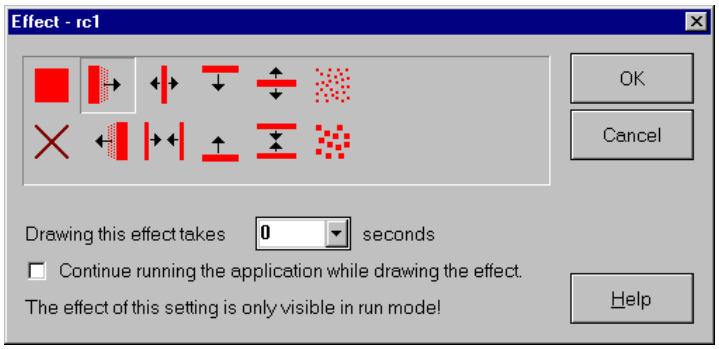
Let a component modify the appearance of other underlying components

These components display color and are dependent on your selection and on the color which is located behind the component. Try to experiment with these settings to become familiar with this attribute.

Format | Effect

You might wish to have certain components appear in a special manner during the course of your application. Often, components are used as temporary storage for values (also known as variables). These components are not intended to be displayed to the user and so you would want them to be invisible at run time. In some presentations, you might also require special dis-

play effects, such as dissolves, curtains, etc. These and more effects can be set using the Format | Effects option.



Define how a component will show up in run mode

These effects are only seen in run mode.

Let's start by describing the effects in the upper row of the Formats | Effects dialog box. The first (solid) symbol makes the component visible. This is also the default effect when a component is placed. The next symbol displays a component from left to right. The displaying of a component from the middle and out, is the third option. The fourth symbol displays the components from top to bottom. The fifth starts displaying from the middle to the top and bottom. A mosaic with small squares is built by the last option in this row. The first symbol in the second row, the X makes a component invisible. This is the most often used effect, aside from visible. The other symbols in that row should be self explanatory. To become familiar with the capabilities, try experimenting with them by assigning an effect to a component and viewing the results.

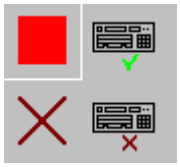
The time required to display these actions can be determined in the dialog box section, Drawing this effect takes x seconds. Select one of the time increments, as desired. Again, experimentation is the key to mastering the possibilities of this feature.

Not all components can respond to the effects attribute. If a component cannot interpret the attribute, it will simply ignore it.

You can set an option to let mindmap continue running the application, while the effect is being drawn. If you do not set the option, mindmap will pause the application, while the effect is being drawn.

Remember that a page is also a component. Therefore, you can also assign effects to a page, so that the next page to which you are jumping, in your application, does not show up immediately, but rather with the selected effect/time combination.

Input fields, list boxes, combo boxes, and data tables offer a somewhat different list of effects.



These effects are available for input fields, list boxes, data tables, etc.

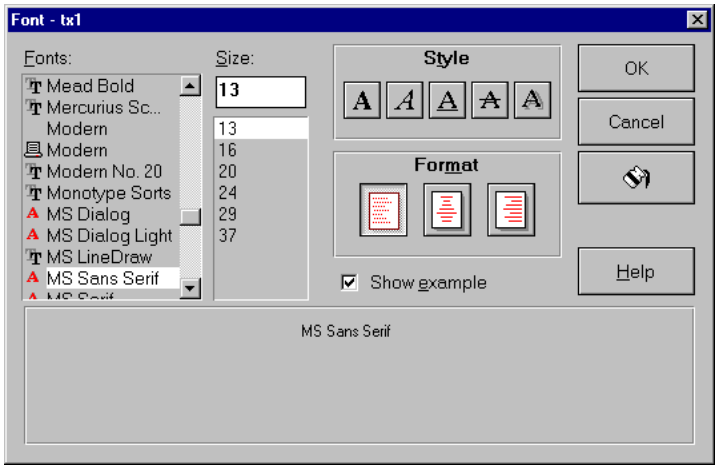
These components can have one of four effects: visible, hidden, keyboard entry enabled and keyboard entry disabled.

If you jump to a component that is hidden, the links on the component are performed and the jump returns to the 'source' component to perform the next link there. If the hidden component is placed on another page, mindmap will not display this page. The links are processed anyway. This is an important feature for the use of 'resource' pages. Such pages may have components that are used for various purposes, such as calculations or menu structures.

Format | Fonts

mindmap does not support font embedding. If you plan on running your application on other PCs, it is suggested you use a standard Windows font. If you need a non-standard font, you must ensure that it is on the target systems, or take care to distribute it along with your application. In the latter case, you must also ensure it's proper installation, as well as deal with any associated license issues.

Various components can include text strings. Therefore, it is necessary to be able to set the font and other characteristics of the font (size, weight, color, etc.) for such components.



Font selection dialog box

Select the desired component, click on Format | Fonts and the font dialog box will open. Select the desired font. The fonts that are available depend on which fonts are installed on your system.

Format | Specific

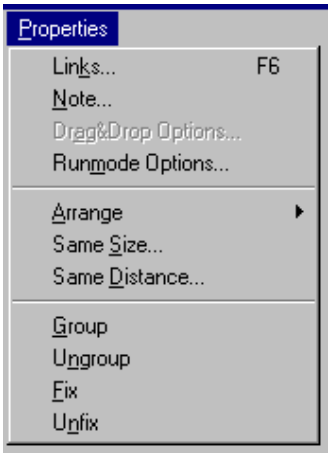
Each component is permitted to register its own private attributes, if it has any. These are specific to the component and thus are not shared with any other

component. All specific attributes are accessible on the menu Format | Specific or through the icon on the attribute toolbox for a given component.

The contents of these specific dialog boxes are dealt with on an individual basis. You can find the necessary information by:

1. looking up the component elsewhere in this manual,
2. addendum's provided with new mindmap-provided components, or
3. with documentation provided with any third party components that you acquire.

Properties Menu

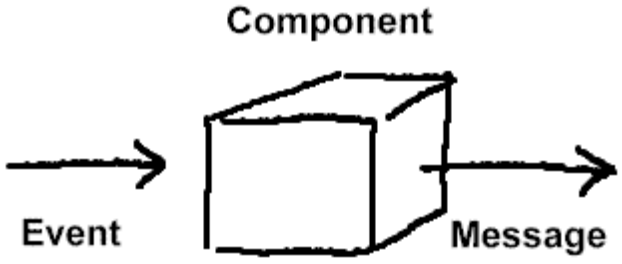


Properties Menu

Properties | Links

This feature allows you to access the dialog box to define links for components. This determines the actual behavior of your application. You can access this feature by clicking on the icon on the attribute toolbox, via the menu option, or by pressing the function key **F6**.

Any component in mindmap, including a page itself, can have links assigned to it.



This graphic illustrates the information flow into and out of a component

In order for a message to be executed, an event must occur. mindmap is a truly event-driven environment. Traditional programmers, who tend to think in terms of flow and procedures, must rethink the paradigm and consider development as defining a string of events and messages.

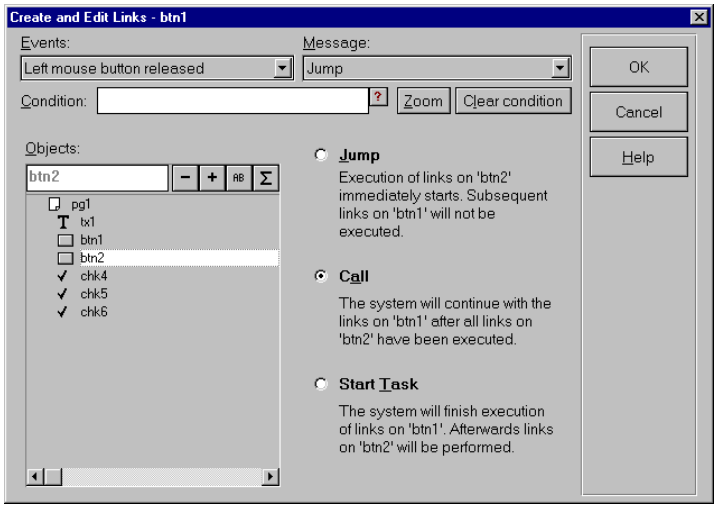
A link is the combination of an incoming event and an outgoing message. You cannot define an event without associating a message. This prevents you from defining syntactically invalid statements.

Most components share a common set of events and messages. A component can also register specific events, which only make sense for that specific component. The same may be done in terms of messages. A component can register a specific message, which becomes available to all other components.

To define the behavior of a component, first define the event on which it is to trigger. Then select the message that it will generate, contingent on the occurrence of the event. You can also associate a condition with a link, thus limiting the execution of the message to certain situations or conditions.

A few examples are:

- When the left mouse button is released, change the fill color of a rectangle.
- When the mouse is moved on top of a check box, display a message on the status bar.
- When the left mouse button is released, jump to page number five, if the amount is greater than 500.



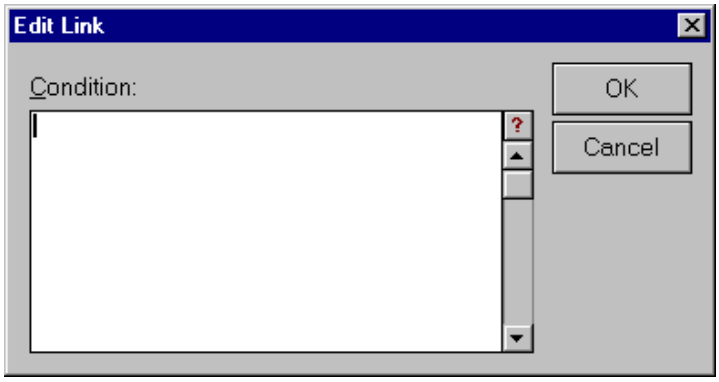
Use this multipurpose dialog box to define and edit links

An example of an event is Left mouse button released. An example of a message is 'Jump to Page 2'. Let's assume you have an application consisting of two pages, with a command button and an input field on page one. The command button contains the link 'When Left mouse button released - jump to page 2'. In run mode, if you click once with the left mouse button when the mouse cursor is over the command button, you will see the command button move down and up. The event Left mouse button released is then handed over from Windows to mindmap. This causes mindmap to jump to page two ('Page 2'), which means that page one ('Page 1') disappears and page two will be shown on the screen.

In addition, you can set a condition. In the example above, you see that the link will only be performed when the condition '`edt1 = 1`' is true. Let's explain this. On page 1 we have placed two components -- one command button (btn1) and one input field (edt1). The message 'Jump' will be performed when the event Left mouse button released is triggered by the component, but only if the condition '`edt1 = 1`' is true. The condition is true if someone has input a '1' into the input field or if another link has assigned the value '1' to edt1.

In the upper half of the Create and edit links dialog box, you see a command button labeled Clear condition. By clicking on it, you can delete the condition formula, if one is in place.

To the right of the condition dialog box is a button labeled Zoom. When you click on it, an additional dialog box pops up:



The execution of a link can depend on a certain condition

If your condition statement becomes large, and requires more space to view than is available in the small field of the Create and edit links dialog box, you can use this larger dialog box to edit your condition. A scroll bar on the right side allows you to edit even longer conditions than would otherwise fit into this field.

By clicking on the button labeled with the question mark, you will get a list of all parser functions that are built into mindmap, along with a short description.

➤ To learn more about functions please refer to page 388.

Also defined in the Objects dialog box is to which component the Jump will lead. Our list of components shows four components, with the name of component 'Page 2' highlighted. Highlighting a component results in displaying its name in the text field in this dialog box.

When you have large applications with numerous components, the list box will not be large enough to show all components at the same time. A vertical scroll bar will be added, allowing you to scroll through a list of all the components in your application.

Toolbox on component list

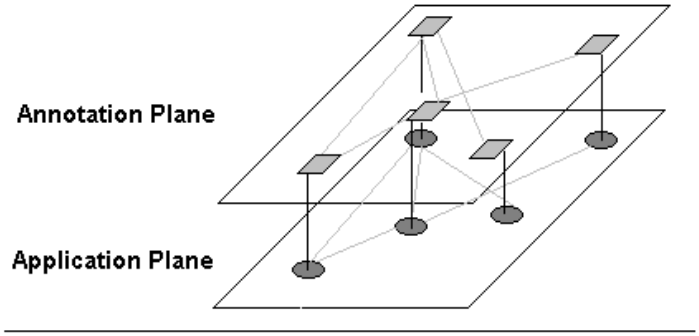
When you open the Create and edit links dialog box of a component, the components in the Objects dialog box will be ordered by pages and within the pages by their creation order. By clicking on the '-' sign, it collapses the list to only display pages. That allows you to quickly find a desired page. By clicking on the '+' sign, the list will be expanded again. If a page has components placed on it, these will be displayed in the order of their creation.

Clicking on the AB button sorts the components in alphabetical order. For example, you need not scroll through all of your component names to reach one that starts with the letter 'Z'. Place the mouse cursor into the white field on the left of the buttons by clicking on that field and key in a 'Z'. The display of the list of components will immediately change and start with components whose names begin with a 'Z', if any exist.

Properties | Note

mindmap offers a feature to help you produce system documentation for your application. This feature is available in the menu Properties | Note, or via the Attributes toolbox that opens by clicking on the component. Here you can type in your annotations (comments, for those of you familiar with coding) and refer to them later.

The idea behind the annotation feature is the goal to physically separate notes or comments from the 'code' in an application, without losing the logical connection. Annotations are tagged onto components. Since the relationship between components is well-defined, via the links structure, the annotations can be mapped on top of the logical structure and connected by means of hyperlinks. This feature allows you to view comments without having to progress through the 'code' to find them. In fact, you may navigate through the entire application by this means. This is especially useful in very large, complex applications.



Relationship between annotations and application

You can print the annotations of one or more components by using the function **File | Print | Objects and annotations**. (See page 37).

When you have selected a component and open the annotations dialog box, you can look at the annotations of other components

associated with it by 'jumping' to those components. For this purpose, you must select one of two lists of components by double-clicking on the text From this object or To this object, in the See also section of the dialog box. All components that have links to or from the selected component, will be listed after this double-click. If desired, you can set a check mark to show only those components which have annotations. After highlighting a component, you can click on the Go to object button. mindmap will deselect the current component, select the highlighted component, and show the annotation of the newly selected component in the Description text field. You can easily jump back to the previously selected component by clicking on the Back button.

Properties | Drag&Drop Options

The Drag&Drop option works with input fields, data tables or list boxes, for example. It allows you to select the option for this component to send or to receive data via drag&drop, by setting corresponding check marks in the dialog box.

This option is also available for some components, via an icon in the Attributes toolbox.

Setting the Drag&Drop option is only necessary if you want the user to perform Drag&Drop operations with the mouse.

Where Drag&Drop operations are performed by links (message), they can be used without setting the Drag&Drop options, because what is dragged and dropped is under control of the application.

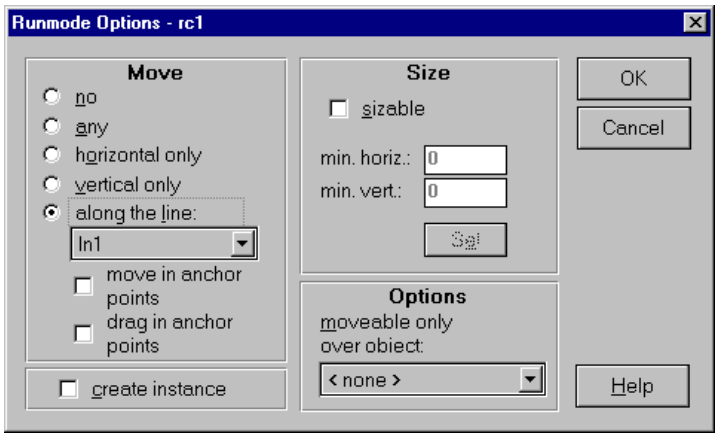
For example, when you place two input fields on a mindmap screen and you select Drag&Drop send data for the first one and receive data for the second one, then you can perform the following operation in run mode: Type in some letters or a number in the first input field. Next, hold the left mouse button down while the cursor stays over the first input field. Then drag the cursor over the second input field and release the mouse button. You will see the content of the first input field is copied into the second input field.

You can use this feature for copying the contents of input fields, data tables and other components to one another. By choosing the appropriate settings, you can even drag the contents of components onto an output page and let the user create his own output in run mode.

Input fields, list boxes, combo boxes, output page components and data tables have additional setting options for Drag&Drop. For more information, see the description of these components in the section about mindmap components on page 117.

Properties | Run Mode Options

mindmap offers some functions that are only visible in run mode. If you select a bitmap, or one of the graphical primitives for example, you will get the following run mode options in the Properties Menu: Move, Size, Options.



Dialog box showing various run mode settings

- Move options are only available for graphic primitives and bitmap components. By default, the Move option has check boxes for no, any, horizontal only, vertical only and along the line. (In the latter choice, you can select a line that you have previously placed on the screen.)
- No will not allow a component to be moved around with the mouse on the screen. This is the default setting.

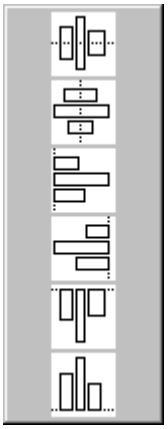
- Any gives the option to move a component to any position on the screen.
- Horizontal only allows the user to move a component only horizontally, starting from it's original position.
- Vertical only allows the user to move a component only vertically, starting from it's original position.
- Along the line allows the user to move a component along a line component, which you have previously placed on the screen. You can select the desired line from the list below the text Along the line. If you have selected this option, you can select the additional options Move in anchor points or Drag in anchor points. This will restrict the movement of the selected component to the anchor points (nodes) of the line. When you have not selected Along the line, these two options are not available and are grayed out.

If you have chosen anything else but no, you can set the option Create instance. (If you have selected no, this option is grayed out). With this option, the user of your application can generate a 'copy' of the component. It can be moved to another location on the screen in run mode, as is allowed by the settings in the move options. The components that have been generated by this method in run mode will disappear when you switch to edit mode.

The Size options allow you to change the size of the component when it is moved. With the Set button you can assign the current size of the selected component to the min. horizontal and min. vertical fields. If you do not want to allow a change in size, check the option fixed.

The Options selection allows you to choose which component in your application will be allowed to move. Select the desired component from the list that you can open by clicking on the button on the right side of the list box.

Properties | Arrange



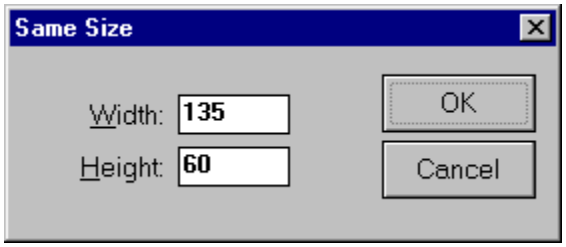
Component Arrange Menu

If you have components on a page, you can arrange them, relative to each other, in six different ways on the screen.

First, you must select the components you wish to arrange. You can then align them horizontally or vertically. The symbols in the Arrange option are described from top to bottom as follows: The first two options arrange the components in a virtual line that goes through the middle of the components. You can align the components vertically left or right, i.e., the virtual line where the components line up is at the left or the right side of the components. The last options allow a horizontal alignment of the components. In this case, the components orient themselves by their upper or lower border.

Properties | Same Size

You can create components of the same size by copying or duplicating them from one component that has the desired size. In cases where you have already created components in different sizes, but which you later want to be a common size, mindmap offers a convenient re-sizing feature.



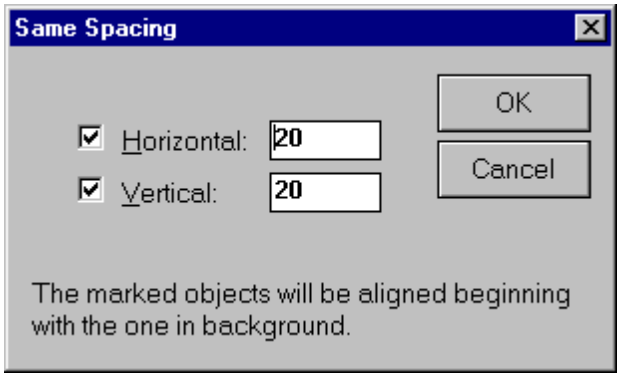
Dialog box showing size settings

To resize one or more components, select the desired components and click on same size. You will get a dialog box with input fields for the common width and height of the selected components. By default, these numbers show the size of the component that has the lowest tab position (farthest in the back-

ground). You can override these numbers by typing in the desired size of your components. The size is given in pixels.

Properties | Same Distance

With this function, you are able to define the spacing between the components you have selected. You need to state the pixel size and whether the distance should be horizontal, vertical or both. You must blank the appropriate field to avoid setting the wrong distance, if it is not desired.



Dialog box showing distance settings

The selected components will be aligned beginning with the one farthest in the background. Do not confuse this with the apparent visual position. Whether or not a component is 'behind' another one depends on its time of creation, not its position on the screen. To view the relative position of a component, switch on the **Edit | Set Tab Order** option.

Properties | Group

If an application has some components that belong together, or if they should have a common link (event), you can group them. That means that you can move them, for example, without changing their internal position or you can use these groups as if they were one single component in the Links dialog.

If you are using Radio buttons, you can use the Group function to determine which buttons will act as one unit. Imagine two groups of three Radio buttons that are arranged on one page. Clicking on one button within one group will not affect the settings of the buttons of the other group, but will reset the other members of its own group.

You can still access the component specific attributes of a single component that is a member of a group by clicking on the group. A pop-up window lists the component members of the group, so that you can get the attribute toolbox of a specific component after selecting the component from the list.

Properties | Ungroup

This command is available for components that are already grouped. After having selected a group, you can ungroup the components of that group. After ungrouping the components, the group itself is no longer available, as it no longer exists. Because any group can be the target of Links, you will be asked by mindmap if you really want to ungroup the components, if such Links do exist.

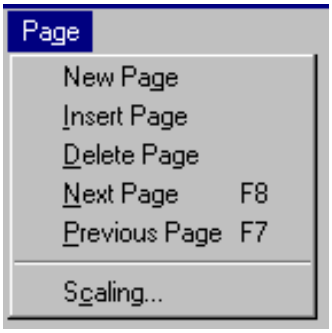
Properties | Fix

Components that never or seldom change their position, or their attributes, can be fixed so that they are always in the same state. To fix a component, select it and click on Properties | Fix.

Properties | Unfix

This command unfixes components that have been previously fixed. As you cannot select a fixed component, unfix effects all fixed components on that page.

Page Menu



Page Menu

Page | New Page

This command adds a new page at the end of an application and makes it the active page.

Page | Insert Page

This command inserts a new page in front of the active page; i.e., if your application has three pages and page two is active, this command will create page number four between page one and page two.

Page | Delete Page

If a page is no longer needed, you can delete it with this command. This command also deletes all components on that page without warning. If the components contain links or are the target of links, you will be asked to confirm the delete function. You will not be asked as long as the components on the page do not have links themselves or are not targets of links.

Page | Next Page

You can move to the next page of your application with this command or by using the shortcut function key **F8**.

Another way to move to the next page of your application is to click on the button with the right arrow in the status bar. With this button, you can also create a new page when no further page exists.

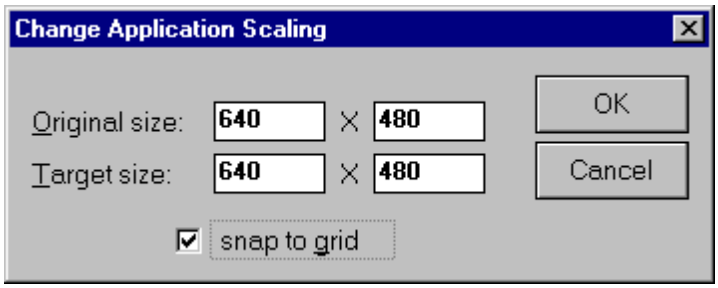
Page | Previous Page

You can move to the previous page with this command or by using the shortcut function key **F7**.

Another way to move to the previous page of your application is to click on the button with the left arrow in the status bar. It's the one located to the left of the field which shows the name of the selected component.

Page | Scaling

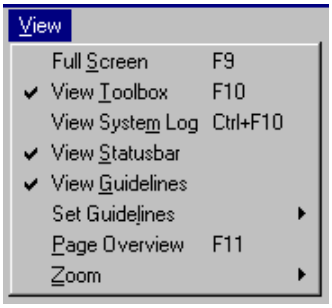
This option allows you to change the scaling of the application. mindmap shows the original size of the application and asks for the target size (in pixels) in a dialog box. You can switch an application that was developed in 640x480 pixels to 800x600 and vice versa with this feature.



Dialog box displaying scaling settings

By default, the components will be scaled proportionally. This can lead to errors due to rounding inconsistencies. The option Snap to grip can be used to adjust the new component size to the actual grid setting in the **File | Preferences | Screen** section. You may find it advantageous to deselect this option, depending on the types of components you have in your application. In particular, care should be exercised when scaling down, as fonts tend to be problematical..

View Menu



View Menu

View | Full Screen

It is sometimes useful to have a complete view of an application page. For example, if you are developing an application in a resolution of 640x480 pixels and your monitor has the same resolution, you will not see the whole page. This is due to the overlapping status and menu bars. The full screen function or the shortcut function key **F9** changes the appearance of the mindmap screen in such a manner that the menu and status bar are hidden. You can now see more detail on your screen.

View | View Toolbox

You can hide or show the toolbox with this command or by using the shortcut function key **F10**. We suggest you use View Toolbox as the default.

View | View System Log

mindmap creates it's own system log, allowing you to see each action that mindmap is taking behind the scenes. It is a visual representation, in a dialog box, of the file, MMERROR.LOG, that mindmap has written. The information that can be viewed here can be helpful in the development process of an application because you can see which error message the parser reports, which database operations are executed, etc.

View | View Status bar

You can hide or show the status bar with this command. We suggest you retain View Status bar as the default.

View | View Guidelines

Guidelines cannot be selected and so the act of deleting guidelines is different from the deletion of other components. Position the cursor on a guideline, press the left mouse-button and then drag the vertical guideline off the left or right side of the screen. The same procedure applies to horizontal guidelines except that you drag them to the top or bottom of the screen.

Guidelines are helpful as you are designing a complex screen layout requiring precise orientation of components. To hide them, use this command. See how to set guidelines in the next section.

Guidelines are helpful as you are designing a complex screen layout requiring precise orientation of components Once you have selected Set Guidelines, you are given the choice of either horizontal or vertical lines. The cursor now changes to a double headed pointer and by clicking the left mouse button, you will create guidelines until you select the pointer in the toolbox to turn this off. The exact position of a selected guideline is shown in the status bar.

View | Page Overview

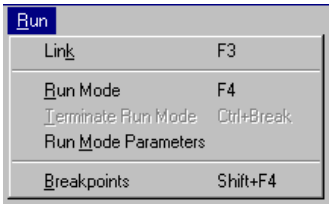
In most cases, a mindmap application has more than one page. To see the pages as small thumbnails in an overview or 'slide show' representation, choose View | Page Overview or use the shortcut function key **F11**. You will see a small representation of the pages and the page names. With a double-click on a specific page, you can easily jump to this page.

View | Zoom

mindmap offers a magnifying glass feature so that you can zoom in, zoom out or switch to the normal view. You can zoom in and out several times depending on the degree of magnification. The feature is available via the menu command View | Zoom as well as an icon on the toolbox.

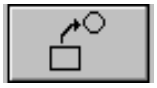
Once you have chosen View | Zoom the cursor will change to a magnifying glass. Press the mouse button and drag the resulting rectangle over the area you wish to magnify. To zoom out again, click on Zoom out or Normal.

Run Menu



Run Menu

Run | Link



The links of a selected component can be viewed graphically, if you choose this option or use the shortcut function key **F3**. A pop up window appears in the lower right corner of the screen and shows all the links of a selected component. A click on another component shows its links.

➤ See also the description of this option in the toolbox section on page 110.

If you are looking for specific links in your application, the link option can also be used. Select one of the Events in the Link dialog box and mindmap will show all components that have this link with a special icon that symbolizes the event. Now you can navigate through the pages of your application and you will see all the components with this selected event.

By dragging the link dialog box away from it's original position (tearing it off), it will stay on top of your screen until closed.

By clicking on the pointer on the toolbox, the Link dialog box will close.

Run | Run Mode

This command switches from edit mode to run mode thus starting your application. You can also switch to run mode by clicking on the corresponding button in the toolbox.

To exit the run mode, you can double-click on the extreme right edge of the mindmap screen. This shortcut will only work if the application does not display vertical scroll bars. This may not work if you are running the application in a window which does not fill the screen, but you can always use the **F4** to switch back to edit mode. The shortcut function key **F4** also toggles you between run mode and edit mode. By switching to run mode, you can work with your application immediately, as there is no compiling in mindmap.

Run | Breakpoints

During the development of your application, you might discover errors or unwanted courses of events. In order to assist you in locating these errors, mindmap offers a debug mode so that you can step through your application one link at a time. Depending on the type of error in the application, you can set a breakpoint on an Event or on a Message. Select the appropriate feature out of the Breakpoints pull-down list and switch to run mode. The application will pause during run mode as soon as the first of the selected breakpoints is encountered. Now you can go through your application step by step and observe what happens. If you choose Step, mindmap will carry out the next link and pause at each subsequent link. If you choose Run, the program will resume and pause only at the next breakpoint.

It is quite common to have events that you wish to occur only if a certain condition is either true or false. You can define these conditions in the links dialog boxes of the components. In debug mode, you can see the condition of the if-button. At the same time, you will see a '1' or a '0' in the evaluation button. '1' signals that the condition is true and '0' that it is false. If you do not see either '1' or '0', you have an incorrect statement in the condition, such that mindmap is not able to interpret the condition. This allows you to test if your conditions have been written in the right syntax and if your application branches to the desired links depending on the conditions.

The Run | Breakpoints feature is very useful in determining the order Links are executed by mindmap and if the application behaves as you want. In order to guarantee that you can trigger on the very first events, be sure to choose Application started as the first event.

The Run | Breakpoints feature offers you various options for debugging your applications by using the five command buttons at the bottom of the dialog box.

With a click on the button Breakpoint on event, a list of all events the component can recognize is displayed. You can choose one event out of the list. In run mode, the application will pause when this event is executed the first time. You can then evaluate the displayed information and proceed by clicking the Step button or the Run button. Notice that you can choose more than one event to break on.

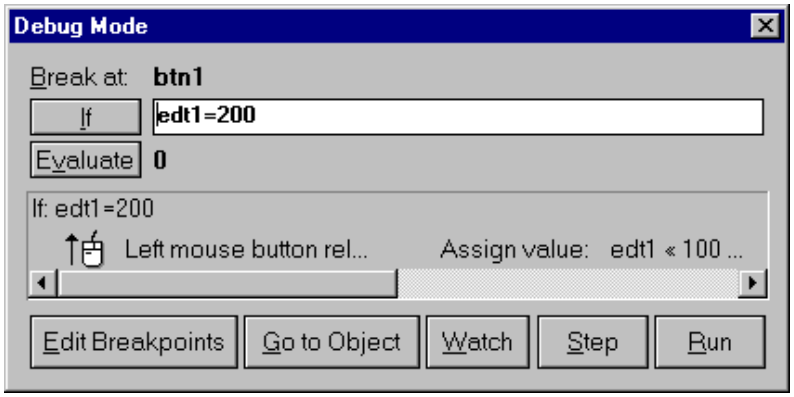
Another method of calling the Debug dialog box is by pressing the **SHIFT+F4** keys.

In addition to choosing an event, you can also choose a message to set a breakpoint on. Just click on the Breakpoint on message button.

With a click on the button Delete Breakpoint, you can deselect an event that is highlighted in the list box, showing the previously chosen events.

How to use the Breakpoints Feature in Run Mode

When you switch to run mode and the first breakpoint is encountered, mindmap will display a dialog box:



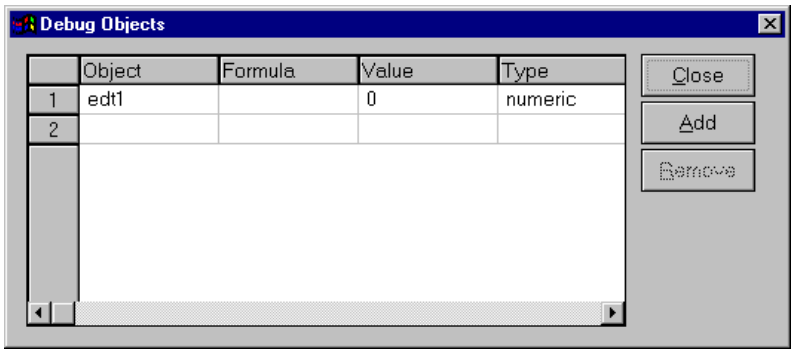
Dialog box representing debug mode

By clicking on the button Edit Breakpoints, mindmap will show a dialog box where you can make all basic breakpoint selections, as described above.

By clicking the button Go to Object, you will be asked if you want to terminate the run mode. If you answer with 'Yes,' mindmap switches to edit mode and will select the current component.

A double-click anywhere in the field where the link is displayed, opens the create and edit dialog box. There you can edit the link. Your changes will be permanently saved with the component.

By clicking the button Watch, a second dialog box will pop up. It shows the Debug Objects dialog box.



This dialog lets you view a component's formula values during debug mode

Here you can evaluate components, their formulas, values and types. To include a component, click on the Add button. This will open a dialog box that lists all components of the application. Choose the desired components by clicking on them and they will appear with their additional information in the Debug Objects dialog box. If you know the component name, you can simply begin typing it. If you click, for example, on the run button of the Debug Mode dialog box, you can watch how values change, depending on your application.

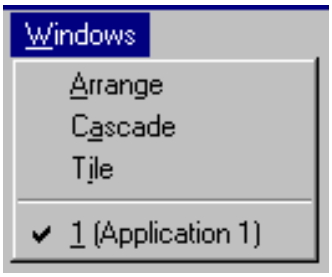
To remove a component in this mode, highlight a component in the Debug Objects dialog box by clicking on the corresponding number in the left most column of the dialog box and then

clicking on the Remove button. The component will be removed from the list, not the application.

You can close this dialog box with a click on the Close button.

Note that the Close button of the Breakpoints message box will only close the display dialog box, but will not terminate the breakpoint debugging mode.

Windows Menu



Windows Menu

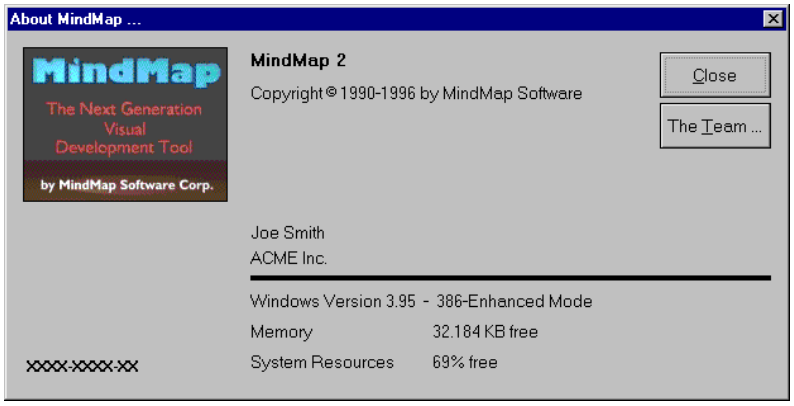
This is a standard Windows menu command. It arranges the windows on the screen and switches between different applications. Please review your Windows documentation for more details on this topic.

If you have two or more mindmap applications open at the same time, you can switch between these applications at any time, by utilizing this menu.

Help Menu

Help Menu

If you click on Index, mindmap switches to the help index. Please review your Windows documentation for more details on using this help system. It works in the same manner as any other standard Windows help system.



mindmap's About box

The About mindmap entry produces the screen above:

In the bottom left corner of this screen, you will find the first 10 digits of the serial number assigned to your copy of the software. This number is your key to various mindmap services, such as online support. The actual serial number consists of 14 characters, of which the last four are considered your personal identification number (PIN). These are not published and should not be revealed to the public.

Chapter 4 Toolbox

The use of the Toolbox

Within this element, you have most of the important and often used components that are needed for developing mindmap applications. We therefore advise you to use **View Toolbox** as the default in the **View** menu, so that you always have quick access to the different components in the toolbox.

Standard toolbox

The toolbox itself can have more than ten components, depending on the modules you are using with the application development environment.

The actual sequence of icons on the toolbox is dependent on the order in which the components are loaded. Normally, the user will not influence this order, but mindmap does offer an option whereby components can be explicitly loaded or, on the other hand, kept from loading. This is controlled by an entry in the MINDMAP.INI file. Under the heading:

```
[Libraries]
Default = *.MDL
```

This will cause every component library (MDL = mindmap Dynamic Link Library), that is located in the home directory, to be loaded. This can be altered, though. Actually, you can even create your own personal load scenarios. To do this, locate the [Libraries] entry in the MINDMAP.INI file. Then enter the desired files, according to the following scheme:

```
[Libraries]
Section1 = DEMO1

[DEMO1]
Default =
Lib1 = MMEDIT.MDL
Lib2 = MMDATA.MDL

[DEMO2]
```

```
Default =
Lib1 = MMBASE.MDL
Lib2 = MMODBC.MDL
Lib3 = MMEDIT.MDL
Lib4 = MMDATA.MDL
```

In this example, only the components contained in the two libraries (MMEDIT and MMDATA) will be loaded. If you later wish to load a different set, then you would merely change the referenced section and the heading [Libraries].

Let us now specify the standard components in the toolbox. Beginning at the top they are as follows:

- Pointer
- Link
- Run mode
- Draw
- Zoom
- Button
- Text
- Graphic import
- Encapsulator (Client / Server)
- Database
- List box
- Data table
- Input field
- Input/output
- Menu
- MCI
- VBX (if a VBX control has been installed).

These components are described in the section about component types on page 117.

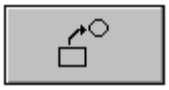
Pointer Option

When you start mindmap you will see an arrow on the screen. This arrow is your standard mouse pointer in edit mode. You can use the pointer to select components on the screen, pick components off the toolbox, make menu selections and execute other mindmap functions. Whenever the cursor changes from the arrow to another shape - because you selected one of the components -- you can switch to the pointer again by clicking on the upper arrow in the toolbox or by pressing **CTRL+Y** simultaneously.

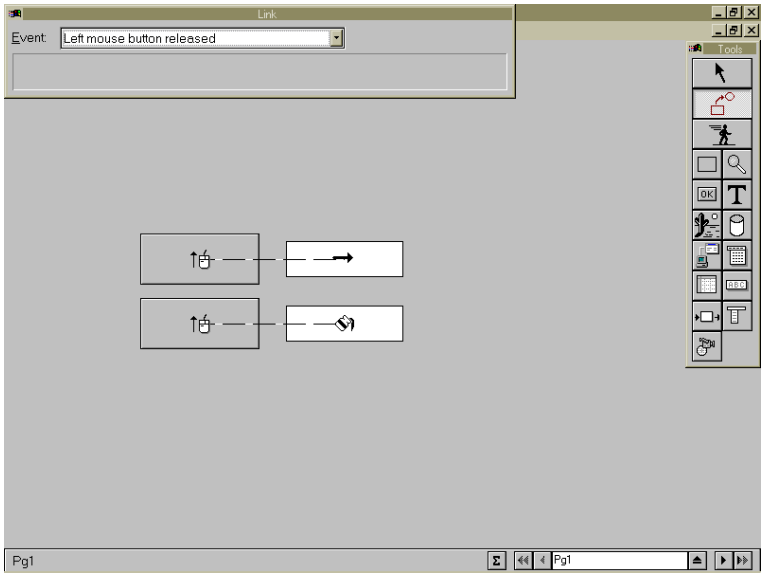
The cursor can take on other shapes, depending on your usage of mindmap and your selection in the toolbox or the components menu. When you click on Objects, a menu opens and shows a selected ✓ pointer. Whenever you have a different cursor and you need the pointer to select and edit components, click on that item. There are two other ways to change the cursor back to its initial setting:

1. Use the shortcut key **CTRL+Y** or
2. Click on the top button of the toolbox which shows the arrow.

Link Option



This tool helps to control and evaluate different events that are connected with different components. Whenever you choose the Link option, a dialog box appears at the upper left side of the screen. Now you can select an event you want to control. For example, Left mouse button released and all the components on the selected page which contain this event will show a special icon symbolizing that event.



This is how *mindmap* visualizes links associated with components

If you select a component in this mode, then the list will display all links assigned to this component, followed by all links which point at the selected component.

The link component function offers a quick overview of selected events and allows you to search for events or to examine the program structure of an application.

If you want to see all events of a component, the link component function is also helpful. Select the event you want to analyze and click on one of the components. You will see that the link component dialog box lists all the events of the selected component. This function allows you to analyze and compare the events and messages of different components very fast and comfortably.

Run Option

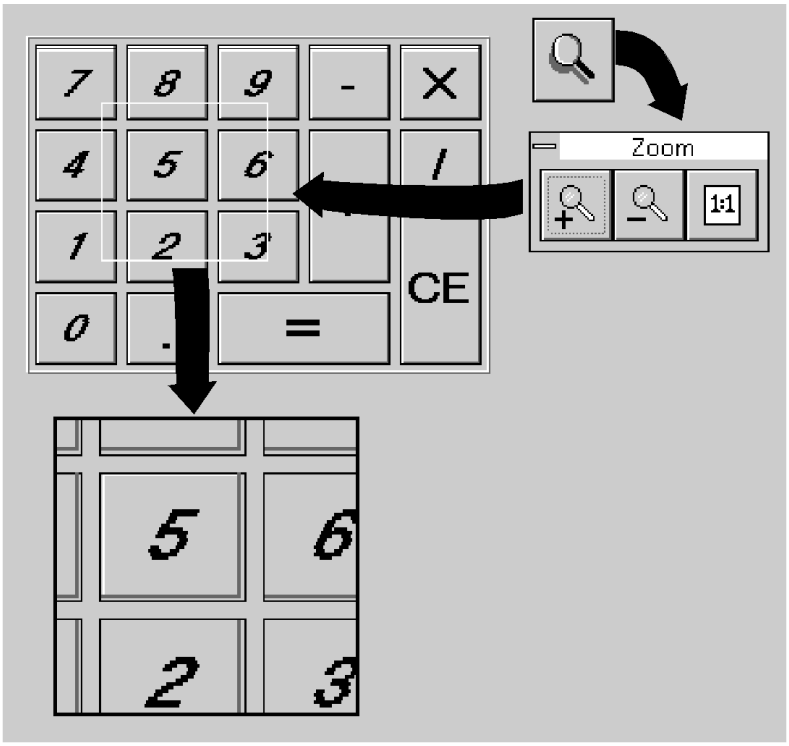
A click on the run mode icon in the toolbox switches mindmap to run mode and makes your application execute. The shortcut function key

F4 allows you to start or to exit the run mode again. The run mode will display the application as the user will see it. The edit mode will, on the other hand, display the application in its construction mode, allowing you to make changes to it.

The run mode helps to check your program without creating an *.EXE-file. You can test certain steps of your development process by switching into the run mode before you save the application. If the result is as you expected it to be, you can save the application. If the result does not meet your expectations, correct your application or close the program without saving, so that you can start from the beginning.

Zoom Option

Working exactly and precisely requires tools that help to position buttons in the right place or to create an output page with exact printing positions.



How the magnifying glass works

The zoom function is a tool that helps to look at things in different degrees of detail. The zoom factor depends on the detail you are choosing, i.e. the smaller the detail you wish to see, the higher the zoom factor must be.

Selecting the magnifying glass within the zoom mode changes the appearance of the cursor so that the cursor becomes a magnifying glass. If you have selected the magnifying glass marked with a plus in the zoom dialog box, you can select a section to detail from a screen. Press the mouse button and pull the cursor over that part you want to zoom in on.

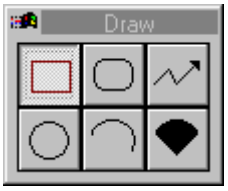
If you want to see the screen in the original size, you can click on the magnifying glass marked with a minus and the zoom factor regresses step by step. The fastest way to get the default resolution is to choose the 1:1 function.

Chapter 5 Component Types

This chapter provides a detailed description of each component and its attributes. Along with the component-specific attributes, all generic attributes relating to the component are described. Reference is also made to those attributes described in previous chapters. If a component registers its own events and/or messages, these are also described in this section.

Draw Component

You can select these components from the toolbox, as well as from the menu. After starting mindmap, you can see the following button on the toolbox, which represents graphical primitive components. If you select this component, you will be offered a choice among the six graphical primitives. After selecting one, its icon is used to represent the group until another one is selected.



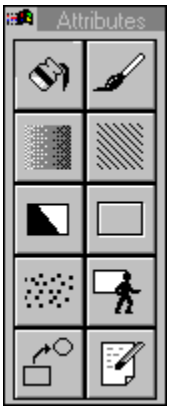
Graphical primitive Toolbox

Clicking on the Draw icon in the toolbox opens a window. This shows six drawing components: a rectangle, a rounded rectangle, a line, a circle, an arc and a pie segment.

Graphical primitives are the simplest components in mindmap. In general, they are used for visual purposes. For example, if you wish to place a colored circle on the screen, you would select a circle and fill it with a color.

Rectangle

Selecting the rectangle within the draw mode changes the look of the cursor. The Pointer becomes a small cross with a little rectangle next to it. With the cross, you can position the rectangle on the screen. You can draw as many rectangles at one time, as you desire. If you want to leave the draw mode, just click on the Pointer in the toolbox. This feature of remaining in the selected mode until you change back to the



Attributes of graphical primitives

pointer is called the sticky mode. This sticky feature is used in the draw mode and for command buttons.

As you can see on the component attribute toolbox, this type of component does not offer any specific attributes.

Special Considerations

Rectangles are very common components in mindmap. Their usage is not restricted to graphical concerns. They are mainly used as:

- placeholders for subroutines,
- variables,
- and other development requirements.

Rectangles and the other graphical components can be sized by grabbing one of the drag marks while pressing the left mouse button. Move the mouse to the desired position on the screen and depress the mouse button. If you select a drag mark in the corner of a component, dragging the mouse will size its width and height. If you grab one of the other four drag marks, the components height or width will be sized. If you press the **SHIFT** key and perform the same action on one of the corner drag marks, the component will be sized proportionally.

Color

A useful feature is the transparent function. You can draw a rectangle, for example, that can be used as a frame for a bitmap. Select transparent as Color for the rectangle and you will see that the surrounding line of the rectangle can still be seen.

➤ A detailed description of the color style begins on page 80.

Changing an attribute of the line is only valid if you have selected No Frame as the border style. This selection is the default setting.

Line Style

The second icon in the attribute box is the brush. This enables you to select a Line type and to define the color of a line. Select a color and change the line width from 1 to 5, for example. Be sure that the rectangle is a little bit bigger than the bitmap. Now you can position the rectangle over the bitmap so that it seems to have the frame of the transparent rectangle.

➤ A detailed description of the line style begins on page 83.

Rounded Rectangle

Selecting the rounded rectangle within the draw mode changes the appearance of the cursor. The Pointer becomes a small crosshair with a little rounded rectangle next to it. With the cross, you can position the rounded rectangle on the screen. You can draw as many rounded rectangles as you wish. If you want to leave the draw mode, just click on the Pointer in the toolbox. Again, this is a sticky-mode component and will remain active until you choose another component or the pointer.

Attributes

The attribute toolbox is almost identical to the one presented for the rectangle. The only difference lies in the fact that the rounded rectangle cannot have a border setting other than No Frame.

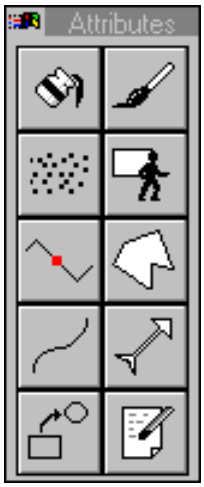
† The available attributes of this component are described in the section Format Menu on page 78.

Line

Selecting the line within the draw mode changes the appearance of the cursor. The Pointer becomes a small cross with a zigzag line next to it. With the cross you can position the line on the screen. A mouse-click starts drawing a line

following your mouse movements. A second mouse-click sets an angle. A double-click with the mouse stops drawing a line. So if you want to draw a simple line, select line in the draw mode, position the cursor at the starting point of your line and click once. Move the cursor to the end point of your line and double-click. Once created, a line can be easily moved and resized by grabbing its handles. This is especially useful in wrapping a line around an object on the screen.

Attributes



Line attributes

Setting a Node

If you wish to define a new node on the line, begin by selecting the line itself. This is done by clicking on it once. Next, locate the position at which you wish to insert a new node. Now activate the attribute toolbox by clicking on the position for the new node. The attribute toolbox will appear. Now select the icon and a new node will appear on the selected line. You can select the node and move it around, thereby defining its position relative to the other nodes.

If you have drawn a polygon, for example, and you want to reduce the number of nodes, just click on one node, activate the Attribute toolbox and select the delete node icon. The node will disappear and a straight line between the next two remaining nodes will be drawn.

Sometimes you will need a graphical component that is not available on the toolbox. With the line component, you can draw the outline of the desired object. Select the line, activate the attribute toolbox and click on the line fill icon. As long as the line contains at least three nodes (which is equivalent to one angle), the two open ends will be connected, resulting in a convex and closed area which is filled with a white color. To change the color, just click on the paint can in the attribute box and select the appropriate color.

The attribute toolbox represented here, reflects the options for a line. It contains an attribute, Lines, represented by a paintbrush. By accessing this attribute, you can set the color, line width and line style. The button displaying a paint can icon permits setting the color fill of a line, only if the line has been closed, turning it into a polygon.

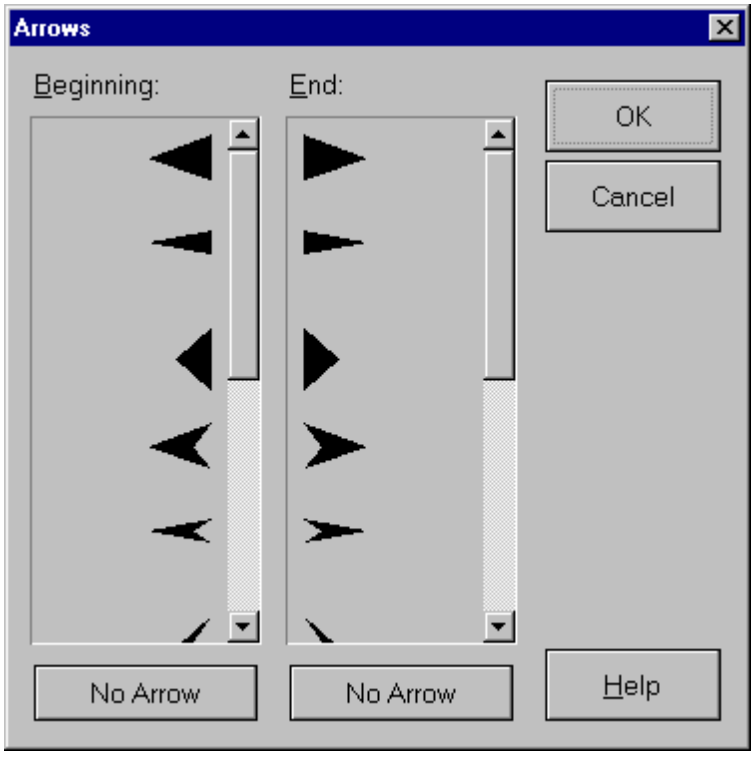
Note that lines that you wish to form a figure need not be connected at the ends. To create a triangle, for example, it is enough to draw two sides of the triangle with one line, and then use the line fill feature.

The fill undo function removes the color filling of lines, so that you get back to the straight original lines, as you had drawn them.

Normally, a straight line exists between two nodes, because the default for drawing lines is straight. mindmap allows you to change a straight into a curved line. Select the line and open the attribute toolbox. Click on the curved line icon. Each straight line segment will now have two crosses in the middle. If you select one of the crosses and then move it, the vector will be displayed as a curved line segment.

This command is available for curved lines only and removes the curved lines created. If you have drawn a curved line and you want it straight again, you don't have to draw a new line. Just select the line and click on the straight lines icon in the lines attribute box.

A line can be changed to an arrow by defining special beginnings and endings for the line. Select a line and open the arrows dialog box from the attribute box.



This dialog box lets you add arrows to a line component

You can use a different color for the line and for the arrow tips. The color of the line is defined via the line attribute icon. The color for the arrow tip is set by using the color fill attribute.

Now you can select the look of your line by choosing the beginning signs and the ending signs. You can choose between ten different signs for the beginning and the end. A total of 10 tips and/or tails are available. If you wish to revert to the previous setting, click on the No Arrow-button.

Consequently, the arrow icon is not available for filled polygons.

Circle

Selecting the circle within the draw mode changes the appearance of the cursor. The Pointer becomes a small cross with a little circle next to it. With the cross, you can position the circle on the screen. You can draw as many circles as

you wish. If you want to leave the draw mode, just click on the Pointer in the toolbox. This component operates in sticky-mode.

The draw mode allows you to create circles and ellipses with the same command. If you have drawn a circle, you only have to pull the selected sides of the circle to create an ellipse.

Attributes

The attribute toolbox for the Circle component is almost identical to the toolbox for the rectangle component. The only difference is again that it does not allow for the setting of a border attribute.

† The available attributes of this component are described in the section Format Menu on page 78.

Arc

Using the Pointer, you can position the arc and determine its size. With the mouse button pressed, you draw a rectangle where the arc is to be located. The size of the rectangle, and thus the arc, cannot be changed. The arc has a marker at the beginning and at the end, and these markers can be used to close or open the arc.

If you position the cursor on the marker of a selected arc, the cursor changes from a Pointer to a pointing hand. Press the mouse button and pull the arc to the desired size and shape, and release the button again. The rectangle specifies the size of the arc and the pointing hand draws the exact arc line, depending on your mouse movement. That means that you don't have to actually draw the arc. You only need to determine the beginning and the end, and mindmap draws the desired arc.

Arc attributes

Attributes

Please note the remarks on the other attributes in the sections above. They apply for the arc, too.

The available attributes of this component are described in the section Format Menu on page 78.

Since this component is actually a line, the color attribute settings are disregarded. If you wish to define a line color for the arc, you should use the lines attribute

Pie segment

With the Pointer you can position the pie segment and determine its size. With the mouse button pressed, you draw a rectangle where the pie segment is to be located. The size of the rectangle and, thus, the pie segment cannot be changed. The pie segment has a marker at the beginning and the end and these markers can be used to close or open the pie segment.

If you position the cursor near the marker of a selected pie segment, the cursor changes from a Pointer to a pointing hand. Press the mouse button and pull the pie segment to the desired size and shape, and release the button again. The rectangle specifies the size of the pie segment and the pointing hand draws the exact pie segment, depending on your mouse movement. That means that you don't have to draw the pie segment. You only need to determine the beginning and the end, and mindmap draws the desired pie segment.

Attributes

The attribute toolbox for the pie segment component is identical to that of the arc. You can set the color attribute to fill the pie segment. The default fill color is set to white.

† The available attributes of this component are described in the section Format Menu on page 78.

Button Component

Buttons are very frequently used components and are, therefore, also placed on the toolbox. After starting mindmap, you will have this icon displayed on the toolbox. It too represents a selection of four components. If you click on this icon, another selection of icons is offered. Clicking on one of these icons will make it the representative icon on the toolbox, until you select a different button.

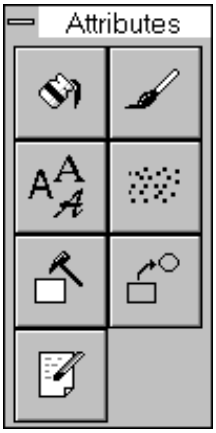
Button component toolbox

mindmap offers four different classes of buttons: command buttons, check boxes, radio buttons and scroll bars.

After making the selection, the cursor will appear as a crosshair with an attached button. Depending on which button was actually selected, one of the buttons will appear as part of the cursor. The cursor is modal (sticky) and will remain in the button draw mode, until you click on the pointer or another icon on the toolbox. This allows you to create as many buttons at one time, as you like.

Command Button

In Windows applications, the command button is one of the most important components for navigating in the application and interacting with the user.



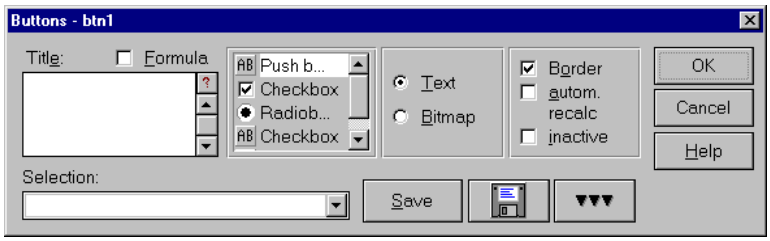
Attributes of button components

Attributes

The attribute toolbox contains an icon offering a font selection, only if the command button has been set to have a text label. This is the default setting. If you change this to a setting where the command button contains a graphic, the font selection icon will not appear on the attribute toolbox.

You can set the color of the command button by using the color attribute. The line attribute permits the definition of the border width of the command button.

You can also select the button's component specific attributes:



Component specific attributes of buttons

If you select Text, you can place text on the button. This is done in the Title field.

If you set a check mark in the Formula field, the content of the title field will be parsed. That means that you can display the result of a calculation as the command button text or the content of input fields (which will be displayed if the title text contains the name of the input field), and the formula check mark is set. This is a very handy feature, if you want to dynamically define the label of a command button. This can also be used to build multi-lingual applications, for example.

Quite often, buttons don't have a text label. In this case, you can include a graphic icon on the command button, which represents the intended function associated with the button. Select Bitmap in the button dialog box. The dialog box expands and shows an additional section where you can paint your own icon, by selecting a color for the left and right mouse buttons. Clicking with the left or right mouse button in the white icon display area paints a pixel in that color that you have assigned to the

mouse button. By clicking on the Delete button, the icon display area will turn white and you can start painting an icon from scratch. It is usually much easier to click on the Selection list. mindmap offers a large list of bitmaps which you can show on a button, ranging from printers and arrows to question marks. The selected icon will be displayed in the icon display area. Here, you can modify the icon to a customized button design. The Save button allows you to save your newly designed icon. Note that you must give a new name to a changed icon before you can save it. Clicking the button labeled with the Diskette icon opens the Windows file dialog box. Here, you can select a bitmap from your system to be shown on the button. Note that the drawing area will display the bitmap converted to the color scheme shown in the left side of the icon display area. If you edit this bitmap, the whole bitmap will be converted. If you leave the bitmap unchanged higher resolution color bitmaps will also be displayed on the button.

The button labeled with the three Triangles opens or closes the icon display area.

The Border check box draws a border around the button. This is a default setting. If the option Automatic recalculation has been checked, then every time the button is pressed, all components on the same page will be instructed to perform a recalculation.

You can set a button Inactive if this is necessary in special situations. The button text or bitmap becomes grayed out and has no functionality.

If you wish, you can let a check box appear as a command button or as a radio button, and vice versa. This might be important in cases where you need to determine the state of a button. Standard command buttons do not retain their state. Check boxes and radio buttons do, but might not be the appropriate representation for what you want to display. In this case, simply redefine the appearance of a radio button or check box to be a command button.

Event	Explanation
Button pressed	This event is relevant where you are using a push button as a check box or radio button.

	This is the default setting.
Button released	see above

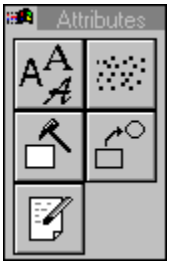
† The common attributes of this component are described in the section Format Menu on page 78.

You can define accelerator keys for buttons. Accelerator keys are used in conjunction with the ALT key to select an action that is normally performed with the mouse. Pressing ALT and the desired character lets mindmap react as if the appropriate push button was pressed.
To do this, include an ampersand (&) immediately before the character you wish the user to use as the accelerator key. Also, consider that on any one page, each such key should be unique.

Check box

Check boxes are a useful feature in the interaction with the user of an application. You can have a range of check boxes in which you ask the user to make certain selections that influence the following steps of the program. Check boxes differ from radio buttons in their multiple function. That means you can have more than one check box selected with a check mark, whereas radio buttons in a group only allow one selection. If you set an attribute for a group of radio buttons, then all members of the group receive the same attribute setting.

Attributes



Attributes of check boxes

The dialog box for the selection of the component specific attributes corresponds to the one for command buttons. The only difference is the selection of check boxes.

Check boxes can only display text, so the option to display a graphic is not available. You can type in text for the check box in the Title field. The text will be displayed in the right side of the box.

➤ The other attributes are described in the section Command Button on page 125.

Again, you can have a check box appear as a command button and vice versa.

Additional Events

➤ Please refer to the description of button specific events on page 247.

The dialog box for the selection of the component specific attributes is virtually identical to the one for command buttons. The only difference is the selection of check boxes as the type.

You can define accelerator keys for buttons. Accelerator keys are used in conjunction with the ALT key to select an action that is normally performed with the mouse. Pressing ALT and the desired character lets mindmap react as if the appropriate push button was pressed.
To do this, include an ampersand & immediately before the character you wish the user to use as the accelerator key. Also, consider that on any one page, each such key should be unique.

Radio button

Radio buttons are used for selections which are exclusive, meaning that making one selection will deselect any other selection. Exactly one selection can be made among radio buttons.

If you do not specify otherwise, all radio buttons on one screen are considered to be logically grouped. Thus, if you select any one of the radio buttons, any other radio button on the page is deselected. If you wish to have multiple groups of radio buttons on one page, then you must define multiple groups of radio buttons. This is accomplished by selecting the radio buttons you wish to have belong to one logical group, and then executing the menu option **Properties | Group**. Create different groups on one page by performing this same operation. Thus, you can select exactly one radio button per group.

Let's explain this by an example: You can paint two radio buttons asking for the gender of a person: male or female. If you want to ask for the age of that special person, you can suggest

different ranges; for example, from 20 to 40, from 41 to 50, from 51 to 60 and from 61 to 70. That means you need four more radio buttons.

To answer these two questions requires six radio buttons. If you would use six Radio buttons on one mindmap page, mindmap would declare them as one group. That means that if you click on one Radio button and select it, any other button on that page that has been previously selected, would be deselected. The solution is to group the Radio buttons that ask for the gender in one group and to group the other four in another group. Now the user can click one button in the gender section and the selection of a button in the age section will not be affected.

You can define accelerator keys for buttons. Accelerator keys are used in conjunction with the ALT key to select an action that is normally performed with the mouse. Pressing ALT and the desired character lets mindmap react as if the appropriate push button was pressed.
To do this, include an ampersand (&) immediately before the character you wish the user to use as the accelerator key. Also, consider that on any one page, each such key should be unique.

The default setting of a radio button is 'not pushed'. Its logical value is thus 'false'.

➤ See also the command **Properties | Group** on page 98.

Attributes

If you select the option Formula in the component specific toolbox, the text entry will be interpreted as a reference to a variable. This means that the text will not be taken literally, but rather as a variable containing the actual text which you want as a label.

➤ See also page 128.

Additional Events

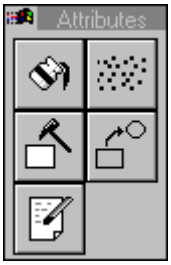
➤ Please refer to the description of button specific events on page 247.

Scroll bar

In the button dialog box, this icon symbolizes the scroll bar function. If you click on this icon, you can create scroll bars. Position the cursor at the starting point of the scroll bar and pull the rectangle, with the mouse button pressed, to the desired size. Depending on the way you draw it, the scroll bar will be either a vertical or a horizontal one.

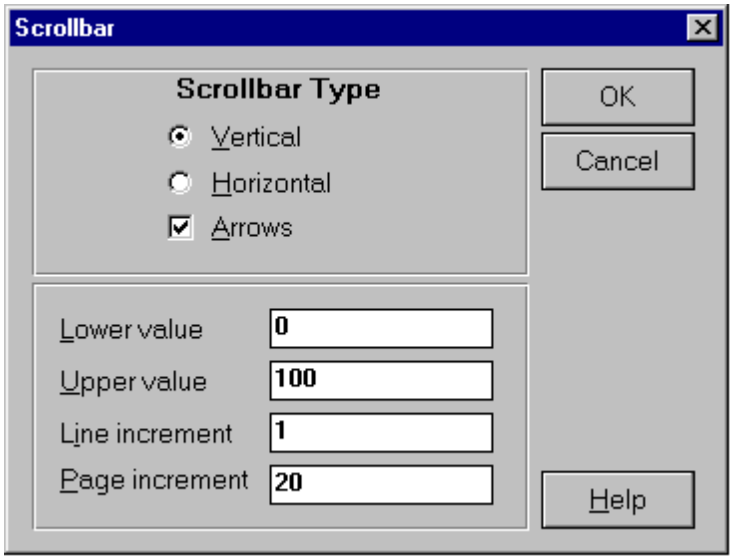
If, when you place a scroll bar, you create a rectangle which is wider than it is high, a horizontal scroll bar will be created. Conversely, if you drag open a rectangle which is higher than it is wide, a vertical scroll bar will be placed on the screen. If you wish to change the orientation, you have two options.

Attributes



Attributes of scroll bar

† Regarding additional attributes, see also the section Format Menu on page 78.



Component specific attribute of the scroll bar

In the component specific attributes dialog box, you can change the scroll bar type from vertical to horizontal and vice versa.

Just click on the type of the scroll bar you want. You can further decide whether your scroll bar has Arrows at both ends or not.

You can specify an Upper and a Lower value. These are the internal start and end values of the scroll bar. The internal difference is always 100 percent. When you move the slider on the scroll bar, the internal value will change as a percentage of difference between the lower and the upper value. If you change the values, for example to years, let's say from 1901 to 1950, the scroll bar movement in run mode results in a recalculation of the internal value. If the scroll bar is moved by 2 percent, the internal value will change by 1 (year). You can assign this value to an input field where the movement of the scroll bar is shown as a percentage interpolation between the lower and the upper value. The value of the input field can then be used for further calculation or for displaying it to the user.

Line increment denotes the step size, when clicking on the scroll arrows on the scroll bar. Page increment specifies the step size when clicking on the scroll bar between the thumb and the scroll arrows.

If you want to use the message Assign value for assigning the internal value of a scroll bar to an input field, type the name of the scroll bar into the formula field of the assign value message in the create and edit links dialog box. Such a link could translate for example to

```
Event: Scroll change. Message: Assign value. Object:
edt1 Formula: scrl1
```

Every modification of the scroll bar with the name scrl1 would then display the internal value of the scroll bar in the input field edt1.

➤ See section **Common Events** on page 244 for more information on other available events.

The following list shows events that are only available in the scroll bar links list.

Additional Events

Event	Remarks
Row up/left	A message is only executed by mouse clicks on the upper (vertical scroll bar) or left (horizontal scroll bar) buttons with the arrows. The bar movement is dependent on the line increment. Note that clicking on the lower or right arrow also moves the slider, but then no message is executed.
Page up/left	A message is only executed by mouse clicks on the upper (vertical scroll bar) or left (horizontal scroll bar) page. (Page means the sector between the slider and the arrow buttons). The bar movement is dependent on the page increment. Note that clicking on the lower or right page (or on the arrow buttons), also moves the slider, but no message is then performed.
Row down/right	A message is only executed by mouse clicks on the lower (vertical scroll bar) or right (horizontal scroll bar) page. The bar movement is dependent on the page increment. Note that clicking on the upper or left page (or on the arrow buttons) also moves the slider, but no message is then performed.
Move	Another way to move the bar is to move the slider with the pressed mouse button in the appropriate direction.
Scroll change	All the above mentioned features are assembled in this event. A message is performed whenever the slider of the scroll bar moves, no matter what caused the move.

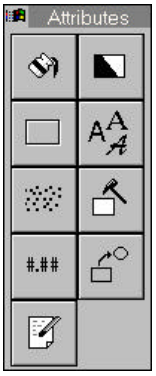
Text Component

This component allows you to draw text fields on any mindmap page. This component can also be found on the toolbox as well as on the menu. It is represented on the toolbox by an icon displaying a capital T.

Selecting the 'T' mode changes the appearance of the cursor. The pointer becomes a small cross with a 'T' next to it. Using the cross, you can create and position a text field on the screen. Once you have placed and/or selected a text component, you will see a vertical line (the insertion point) inside the text component. It will be in the first column/first row. This line symbolizes the cursor inside the component and displays the current position.

Similar to the command button component, you can include an ampersand (&) within the text. The consequence will be that the letter immediately following the ampersand will be displayed with an underscore. Since text components do not allow user input at runtime, it does not make much sense to grant them focus. Accordingly the focus is passed to the next component in the tab order. You can use this feature to position the cursor in an input field by assigning an ampersand to a text field which is used as the label for the input field.

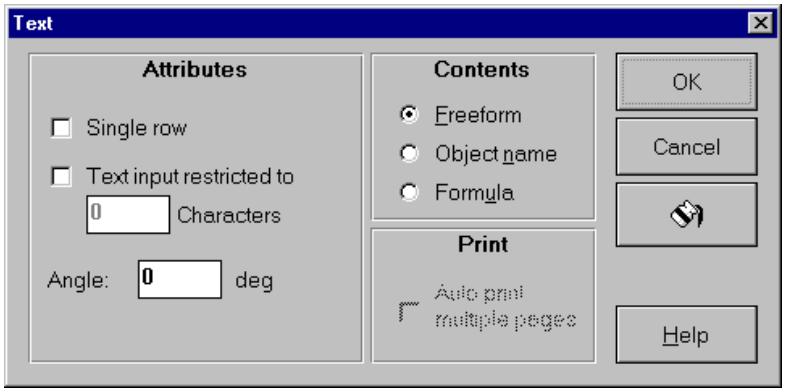
Attributes



Attributes of text component

The common attributes of this component are described in the section Format Menu on page 78.

Component Specific Attribute



Specific attributes of the text component

In the component specific attributes dialog box, you can determine whether your text display is restricted to a single row. You can also restrict the text input to a defined number of characters.

An additional feature for showing text on the screen is the Angle function. Here you can specify the number of degrees at which the text is rotated. You will need to experiment a little bit to understand the underlying logic of this feature. The angle has different effects, depending on the angle and the chosen text font.

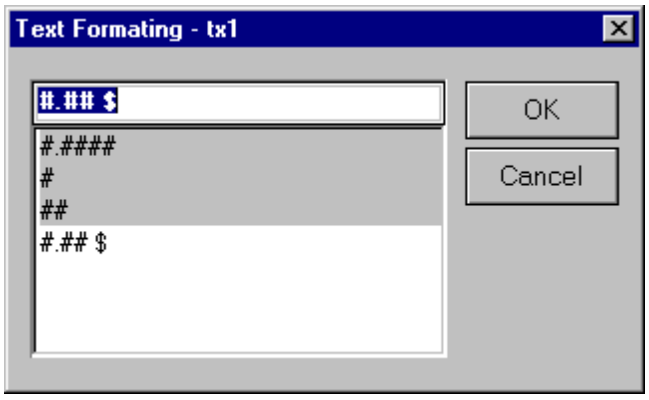
The text angle function only works with true type fonts. If you select a non true type font and want to angle the text, mindmap will ask you to select a true type font. Consider the fact that the Angle influences the size of the text field. That means a horizontal text block may fit in the text field, but if you select an angle of 45 degrees, for example, the text that does not fit in the text field will be cut off. Make certain that the entire text block appears as you want it and adjust the size of the text field relative to the angle/font combination.

The text fields appear as Freeform by default. If you change the content to Object name, mindmap will display the name of the component in the text field. If you select the option Formula, then you cannot enter text, as you are able to do when Freeform is selected. If you wish to have text appear in the text field, you must either use an Assign link, revert to the option selection Freeform, (see also **Assign Value** on page 258) or enter the text into the parser window (the button displaying a in the status bar).

An additional option is available for text components placed on output page components. This permits a text field to wrap to a subsequent page, if its content -- which is defined at runtime -- is larger than the space provided by drawing the original text field on the output page component.

Format Attribute

If the formula of your text field is a numeric value rather than a string (such as a number that is not enclosed in quotation marks), you can specify a format for the text field component. If you do so, the following dialog box appears.



Format attribute dialog box of text component

This dialog box permits the definition of a specific format for numerical values. For example, you can determine where the decimal point is to be placed. You can also have a character, such as a dollar sign '$' included.

Additional Events

No additional events are registered.

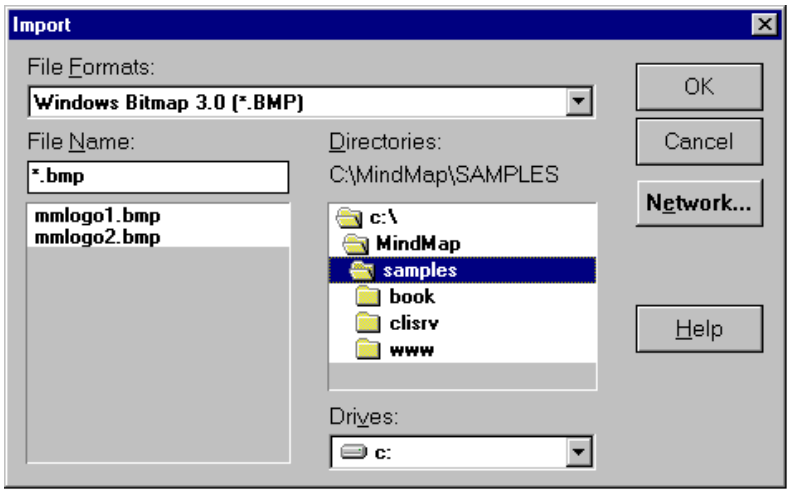
Special Notes

The **CTRL+Ins** and **SHIFT+Ins** keys are used to copy the selected contents of a text object into the clipboard. To copy the text component itself to the clipboard use the **CTRL+C** and **CTRL+V** keys. Note that in this case the contents in the clipboard are visible only to mindmap.

When in edit mode, you can copy the content of the clipboard into a text field and vice versa. This is only possible if the text field has been set to freeform. Select the text field you wish to copy from and press **CTRL+Ins**. This copies the content of the text field into the clipboard. If you now wish to place the content into another text field, select it and press **SHIFT+Ins**, which is the shortcut key for inserting from the clipboard.

Graphic Import Component

The graphic component allows you to include various graphic files as part of your mindmap applications. Once you have selected this component from the toolbox, the following dialog box appears.



Select a format and a graphic file to be imported into an application

Only 16-bit versions of graphic import filters which follow the Lotus/Microsoft/Intel specification are recognized.
When importing images, you may wish to use the standard integrated mindmap filters, since these support dithering.

In this screen shot, the list containing the supported file formats has been expanded. The formats in this list, which are displayed without the asterisks in the front, are standard graphic file formats mindmap supports internally. Formats listed with an asterisk in the front are only available if you have installed other Windows programs such as Microsoft Office, Lotus SmartSuite or a similar program belonging to this group, and have selected Import filters at the time of your original installation. mindmap recognizes these filters and will use them to read files that you may wish to import.

Before you import a graphic, select the File format and then load the graphic. This may take time, depending on the size and the format of the graphic. Next, the cursor changes to a paint can and you can position the graphic on the page. There are two methods for placing the graphic. You can simply click once on the left mouse button. This will place the graphic in its original size and is the preferred method. The mouse position will define the top left corner of the graphic. You can also size the graphic when placing it, by pressing the left mouse button and dragging the mouse to represent the desired size. You can always change the size of the graphic later by either interactively sizing it, or by accessing its specific attributes and setting the desired size in terms of pixels.

If possible, you should avoid sizing a bitmap, as this might lead to undesirable visual effects, due to the loss of pixels.

If a smaller representation of a graphic is required in an application, we strongly recommend that you use one of the various available graphic tools to re-sample the image. Due to the fact that, by default, mindmap saves images in the application file, you can dramatically reduce the loading time of your application, if you reduce the size of the image.

This applies not only to the geometrical size of an image, but also to its color resolution. There is no need to import a 16-million color image, if it only uses 16 standard colors. Try to convert an image to a minimum amount of colors.
There are may shareware and/or freeware products available on the market to perform these tasks.

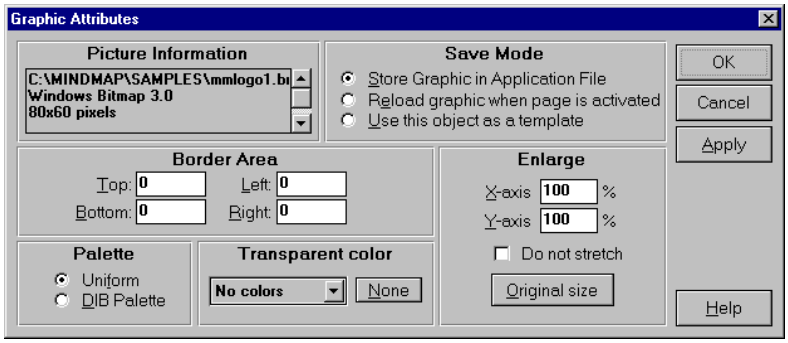
Attributes

Attributes of graphic component

The attributes Color, Lines, Fountain Fill and Fill Pattern, all relate to the frame surrounding the graphic. The default width of this frame is zero. In order to set a frame, you must crop the graphic. This is accomplished by moving the mouse to a corner of the graphic. Press the **CTRL**-key and move the corner with the mouse. The cursor will change from the normal pointed index finger to now look like two right angles facing one another. Cropping retains the original size of the graphic and defines a new viewing mask. If you thus make the graphic larger than its actual size, it will receive a frame. If you make it smaller than its actual size, you will crop part of the graphic, while retaining its aspect ratio.

† The common attributes of this component are described in the section Format Menu on page 78.

Component Specific Attribute



The specific attributes of the graphic component

Graphic Information, in the component specific graphic attributes dialog box, gives you details about the graphic. You can see the name, the path, the filter and the size (in pixels) of the graphic. These parameters describe the imported graphic file as it will be saved into the application file. It does not reflect the actual settings of your graphic adapter.

If you import a true color graphic file on a computer that has a 256 color video adapter and move the application to a computer which is able to display 16 million colors the image in the application will automatically be displayed at its best resolution.

Save Mode offers three different ways to save the graphic. Which save mode is the best, depends on your application.

- As the default, mindmap suggests Store Graphic in Application File. Each graphic stored in the application file will enlarge this file. Keep in mind that the size of graphic files is generally large and this will tend to dramatically increase the size of your application.
  Also note that mindmap saves all types of pixel-oriented input formats as if they were bitmap files. Input formats using extreme compression algorithms (TIFF, JPEG) may increase the size of the application file dramatically.
- Reload graphic when page is activated is a good possibility, if you can be sure that the graphic will actually be available at the specified location when the application is eventually executed. The benefit of this is that the application file does not contain the graphic file(s) and is thus generally much smaller. Deployment of your final application will require careful thought, if you choose to employ this method, but it has several advantages.
- If you have a page in your application where different graphics are to be shown (at one and the same screen position), you can use the function Use this object as a template. By using drag&drop between an input field and a graphic template, you can assign a new graphic to this template. This allows you to have a single graphic component for graphics display in your application and, via the drag&drop function and the template option, many graphics can be shown. For example, imagine a list of graphics. Whenever the user selects a file name, the graphic is displayed on the screen.

Border Area refers to the frame created by cropping a graphic. You can also create it by making the appropriate settings.

Palette (applicable only on 256 color video adapters): Here you can select whether a graphic uses its predefined DIB (Device Independent Bitmap) palette or a mindmap-assigned uniform palette to show the graphic. If you want to show more than one graphic in 256 color mode on one page, you have to choose the

uniform palette and the graphics will be shown with a common palette. If you choose the DIB palette, the graphics will switch between different palettes and the palette of a graphic which has the focus will be applied to other graphics on that page. Sometimes it may be better to use the DIB palette, as the graphics will have a better display quality. Due to the wide variety of possible graphics combinations, you must experiment with your individual situation for the best choice of palette.

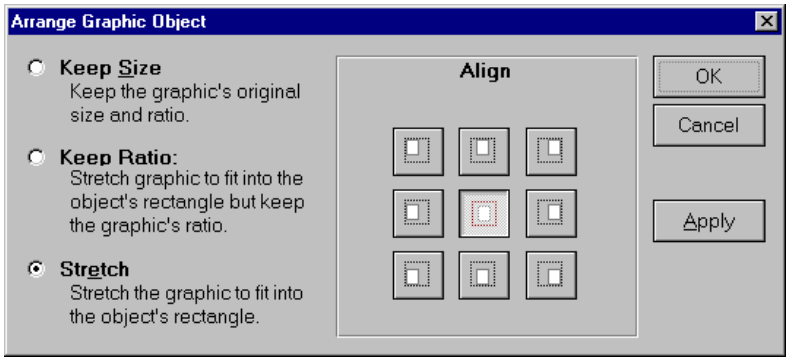
mindmap uses a technique called dithering, to map the color pixels of an image to a common set of colors (a palette). The benefit is that multiple images can be displayed simultaneously on the same page. The disadvantage is that this technique increases the load time of an image.

Transparent color gives special effects to graphics. You can select transparent as the color, so that the background of the graphic will show through. Note that this is not applicable for images which consist of more than 256 colors.

You can Enlarge the graphic, as measured in percent. mindmap re-sizes the graphic so that it will become bigger or smaller. You can select that the graphic will not be stretched when you change the size. This means that its height/width ratio will not change if you type in another size in either the X-axis or the Y-axis fields. Clicking on the button Original size sets the X-axis and Y-axis values back to 100 percent and the graphic will be shown in its original size. Please note that these settings depend on the Arrange Graphic Object settings.

Arrange Graphic Object

Especially when you work with templates (see the component specific attribute Save mode), it is necessary to specify how the graphics should be placed in your application.



Select how a picture is to be aligned within its bounding frame

Keep Size: This function keeps the graphic's original size and ratio. Whenever you have different sized graphics, whose size has to be fixed, you should set this radio button to on. Your graphic will not be changed and will be displayed in its original size.

Example: If your graphic is larger in size than the template and you have clicked keep size, the template will be overlapped by the graphic.

Keep Ratio: Whenever you drag&drop a graphic to a template and the size of these components vary, you should click on keep size or keep ratio, depending on your graphic. The function keep ratio stretches the graphic to fit into the component's rectangle, but keeps its ratio.

Example: If your graphic is larger in size than the template and you have clicked keep ratio, mindmap reduces the graphic to fit into the template. The graphic becomes smaller in size, but keeps its ratio.

Stretch: Sometimes it is very useful to change the size of a graphic, so that you can use one graphic on different pages in different sizes. You can also use this feature if you intend to display various graphics in the same screen position and wish to have them all appear in the same frame. In that case, one can use one big graphic and reduce it to a smaller one on another page. (This results in a better quality than blowing a small graphic up to a big one). This function stretches the graphic so that it fits into the component's rectangle.

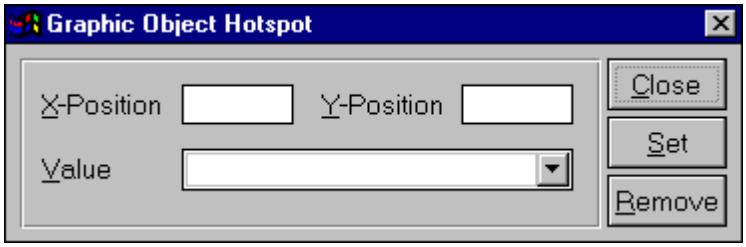
Example: If your graphic is larger in size than the template and you have clicked stretch, mindmap re-sizes the graphic to fit into the template. The graphic becomes smaller in size.

Align: Whenever the size of the template differs from the size of the graphic which is to be displayed, you should set the alignment. You can choose between nine different positions from the upper left corner of the rectangle to the lower right position.

The function Apply is used for testing the Alignment. This way, you can check which function best fits your needs.

Graphic Component Hot spot

mindmap offers the possibility to put messages on top of graphics. This is very useful for interaction with the user. If you want to access certain parts of your graphic, you may be able to use the hot spot function.



The graphic component hot spot

After you have opened the graphic object hot spot dialog box, you can click with the left mouse button on a part of your graphic. Clicking on the graphic while the Hot spot dialog box is open, changes the color of the hot spot and displays its position and value in the window. In the dialog box you can see the X- and Y-position of the hot spot. Select all the desired hot spots you want and click on Set each time. By clicking on Set, the corresponding coordinates are entered in the list. If you wish to assign a character string to a coordinate (such as a city name on a map), enter the string in the Value field and then click on the Set button.

In the Value list box, you can see the coordinates or the character string assigned to the hot spot.

Example: If you have imported a graphic of a map with a number of cities on the mindmap page and you want to produce special information about these cities in Run mode, click on one of the cities to create a hot spot in Edit mode. Now you can name the hot spot via the value in the list box. This value can be assigned to an input field. Whenever the user clicks on one of the hot spots in Run mode, the given name of this hot spot is shown in the input field. You can then connect this input field with a link, for example, to a list box or another component that can display additional information corresponding to the selected hot spot.

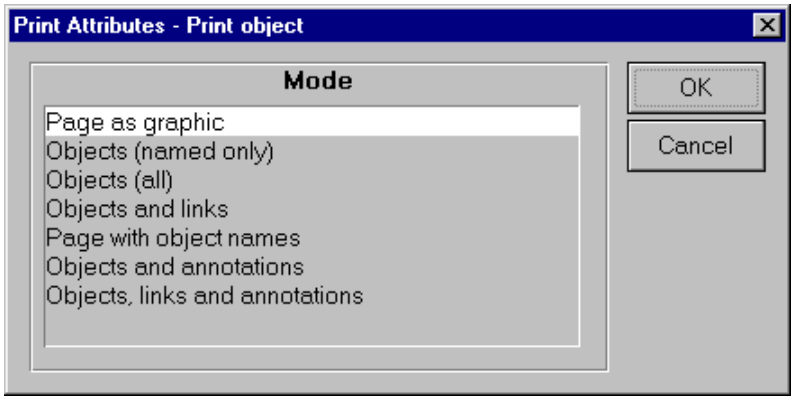
When the user clicks on the hot spot, the component itself will receive the value associated with the hot spot. The selected hot spot will display in green; all others in red. If you select one hot spot, all others are deselected.

Additional Events

No additional events are registered.

Printer Component

This component is only accessible through the menu and is grayed out in 'normal' mindmap applications. It is only used in printer layout files. The printer component is used for printer templates only. The printer component specifies the layout of the printer template. This template is stored in the file, STANDARD.MMP. The printer component is a helpful tool for documenting and analyzing your application.



Printer component attribute toolbox

† For further information concerning common attributes see the section Format Menu on page 78.

Attributes of printer component

Component Specific Attribute

mindmap offers the following modes for this component:

Mode	Description
Page as graphic	Prints the page as it is displayed
Objects (named only)	Prints a list of all components with their names
Objects (all)	Prints all components with name and symbol
Objects and links	Prints the components and links
Page with object names	Prints the names of the components of one page
Objects and annotations	Prints the components with name, symbol and annotations
Objects, links and annotations	Prints the page, the links, the components and annotations.

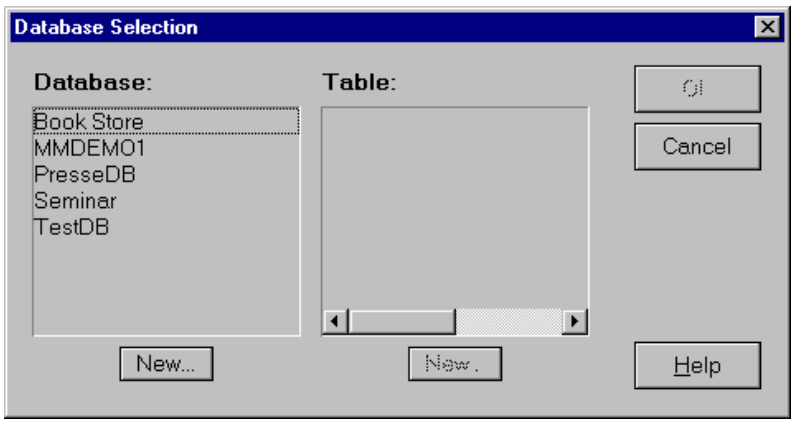
Database Manager Component

A powerful feature of mindmap is the database component. You can connect a database with the Windows ODBC (Open Data Base Connection) driver. This component permits you to access existing data sources and, in some cases, to create new sources.

Please be sure that the necessary database drivers are installed and correctly configured. These are either already installed or you opted to install them during the mindmap setup routine. If not, you can install the drivers using the ODBC Administrator tool, which should be located in your \WINDOWS directory.

If you click on the database icon in the toolbox or the menu option **Objects | Database Manager**, a dialog box opens where you can select the database driver, normally ODBC. mindmap offers you an option to install additional modules. These modules allow you to access other systems and data sources, which might not be available via the ODBC interface. If you have installed supplemental mindmap drivers, data sources accessible through these drivers will also be listed.

If only one mindmap database driver is installed, mindmap automatically proceeds to the Database Selection:



Select a database and a table to work with

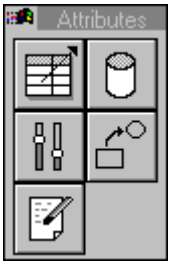
In this dialog box, you can select a Database and the corresponding Table. The tables will be listed as soon as you select a database. Please note that, while your application may require multiple tables to operate as you intend, mindmap allows you to place only one table at a time. You may have many tables placed and open at any time.

Please note that the names which appear in the ODBC data source list are not necessarily identical to the actual database names. The ODBC Administrator program allows you to assign names of your choice to database files.

The New buttons allow the setup of new databases and tables. In the first dialog box, you select the special ODBC driver for the database, e.g., Access Data (*.MDB) or dBase files (*.DBF). The following dialog box helps to choose the appropriate database or to create a new one. These dialog boxes are described in more detail in the documentation dealing with the ODBC Administrator. The ODBC Administrator can be accessed through the Windows Control Panel or through the mindmap program manager group. Please note that not all database drivers permit you to create new databases. In addition, depending on the database and your company policy, you may not have creation permissions. Please consult your database administrator for more details.

After you have selected the database and the desired table, the cursor changes to a cross with the database icon next to it. You can position the database component anywhere on the page. The indented component display shows the driver, the name of the database or data source, and the table name. The Database wizard dialog box then appears. You may want to use the Wizard to automatically set up connected input fields or a data table, as well as some buttons. We assume here that you just click OK.

Attributes

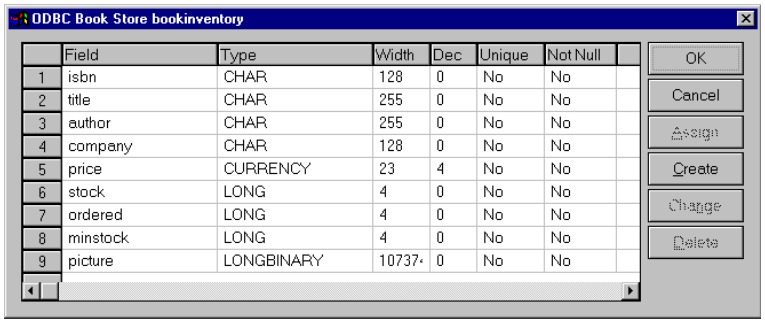


Attributes of database component

The common attributes of this component are described in the section Format Menu on page 78.

Table Structure

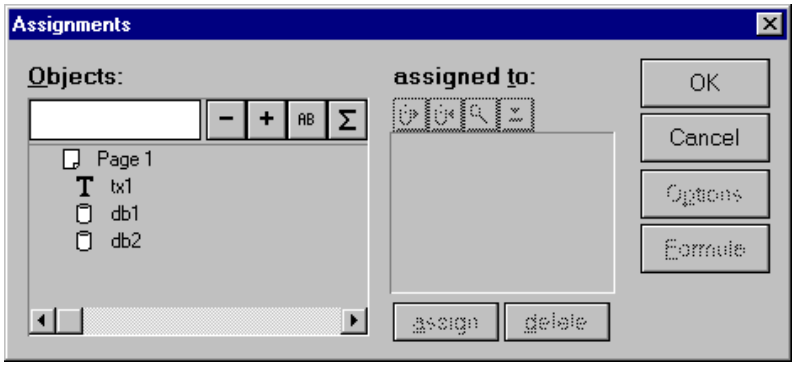
A click on this attribute of the database component opens a dialog box representing the structure of the database table. All fields are displayed with their definitions, as defined in the database itself.



Field definitions of a database table

This component is only visible in edit mode. Once you switch into run mode, this component becomes invisible to the user.

You can assign these fields to mindmap components such as input fields or data tables, for example. Click on a row number (first column) in the table (representing a field) and the row is now highlighted. Select Assign, so that the Assignments dialog box appears.

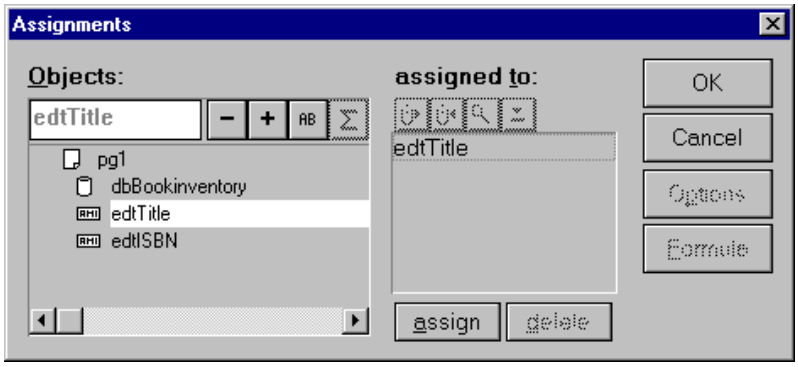


Assign a particular database field to components within your application

The simplest and, therefore, most often used technique for assigning components to database fields is the so called rubber band method. Open the table attribute of the desired database component that has already been placed on the page. Click on the row number of a database field (*the row will thus be highlighted*) and keep the Left mouse button pressed. Now, drag the cursor to the component that will be assigned to the field. As soon as the cursor leaves the dialog box, you see the rubber band at the cursor. Position the cursor over the component and release the mouse button. If you successfully established the connection, the associated component name can be viewed in the last column of the assignment dialog box. All assigned components are listed there. Normally, you must scroll over to the right to view this column.

In the left side of the Assignment dialog box is a list with the components of your application. Select the component by clicking on it and press the assign button or just double click the component. Now you can see the component name in the assigned to list.

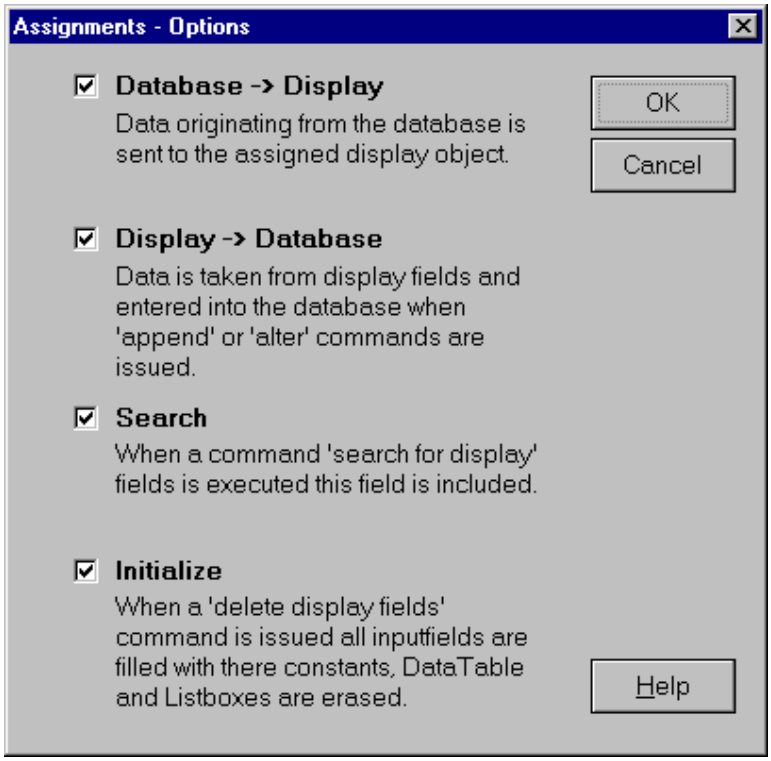
You can assign multiple components to one field. All of them will then be listed. Each component can have individually set options, but only one component can have write access. mindmap helps to prevent conflicting write access by prompting the user to choose which component shall have write access.



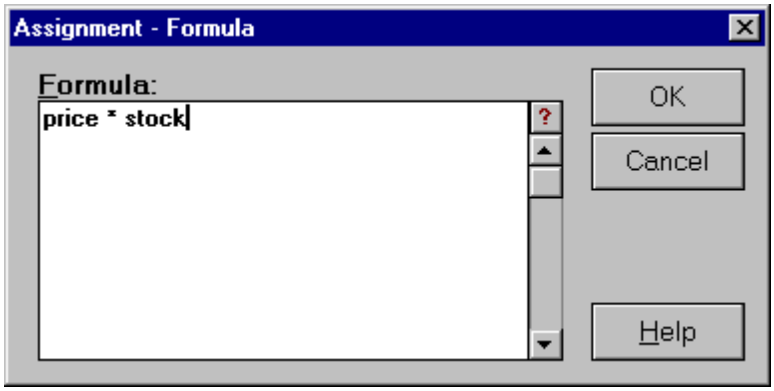
The database field has been assigned to an input field

If you select the component in the assigned to list of the Assignment dialog box, four small buttons above the list will be activated. A click on one of them will change the assignment option. You can reach them all together via the Options button. The following table shows the assignment options. The buttons represent the settings in the following dialog box, which were called via the Options button.

Function	Icon	Description
Database ➔ Display		Data originating from the database is sent to the assigned display component (read access).
Display ➔ Database		Data is taken from display fields and entered into the database when insert or update commands are issued (write access).
Search		When the command search for display fields in the message part of the link 'Database' is executed, the database searches for the contents of this field. This means that the contents of the WHERE clause in the SQL statement will be included. Please note that for the input field component, you may select from a list of predefined search methods.
Initialize		When a Clear display fields command is issued, all input fields are filled with their given preset values. The contents of data tables and list boxes are erased.



Control the data flow to and from the database



The result displayed in a component can be calculated from various database fields

The Formula button in the Assignments dialog box contains the name of the database field, by default. Here, you can type in a formula that will be parsed The database component will then evaluate the given expression for each record retrieved from the database and assign the result to the corresponding mindmap component.

You may want to use this function, if the database source cannot perform a similar task. To increase the overall performance of database access, you should attempt to let the database perform calculations like this. This is especially true for client/server SQL database systems.

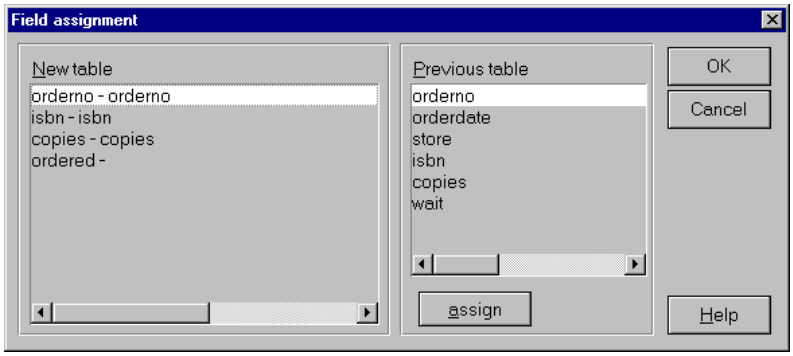
Please note that you cannot update into the associated table field, regardless of whether you let mindmap do the calculation or the database system.

If you want to Delete the connection from the database to one of the components in the list, select its name and the Delete button will be activated so that you can delete the assignment.

Database

You can use this button to replace one database with another one. This may become necessary if the structure of the database tables has changed. This works as described before with a new database, except that a field assignment dialog box appears where you can assign the fields of the New table to the Previous Table. When you load the new table, mindmap automatically suggests new assignments. This is only possible if both tables (the one being replaced and the replacement) have corresponding field names. You can also override the definitions or make the necessary assignments manually, should the field names be different.

Example: This function is quite useful in the case where you are developing an application on your PC and running against a PC-based database. You then want to move it to the production system, perhaps an Oracle or Informix database, and deploy it in the organization. This is where you would re-map the database tables to reflect the new database and its tables. If you have the same field names in the tables of both databases, they will map across automatically. If not, you will have to do some manual mapping of the assignments.



Map the fields from a new database table to the already existing component

Database mode

An additional attribute of the database component is the Database mode.

Select how *mindmap* works with the database

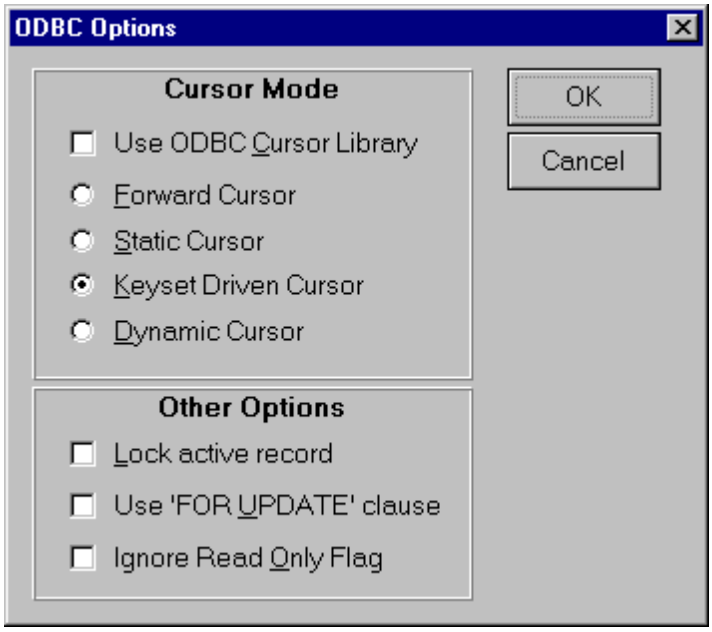
- Show first record after opening database:
  If this option is checked, mindmap will automatically select the first record that the database system has found in the appropriate table and display its contents in the associated input fields. If a data table has been connected to the database component, all records in the table will be displayed. Obviously, if the table contains a huge amount of records, you may want to clear the check box for this option. Depending on the kind of database system you are using, it

may take a very long time to create a result set containing all records in the table. Better is to clear this option and create a new result set of a reasonable size by either issuing the database message Search for Fields, or directly executing a SQL statement, if supported by the database.

- Commit automatically:
  This option is supported only in the case of transaction oriented SQL database systems. These databases allow you to 'undo' the execution of certain database commands (such as insert, update or delete). This is called a rollback in the terminology of database systems. Writing multiple transactions persistently to the database file is called a commit operation. The commit automatically option specifies that changes to a database table are automatically committed. Consequently, rollback operations are not applicable in this case.

Note that ODBC database drivers like dBase, Text or Excel do not support commit/rollback commands. Only Microsoft Access and most of the available SQL databases support this feature.

You can set More Database Options with the appropriate button.



Control various ODBC Options

Additional Events

In addition to a number of standard events, the database component has the following additional events:

Event	Description
Record loaded	As soon as a record is loaded, this event is triggered.
Refresh data fields	A refresh of data fields triggers this event.
Delete failed	This is a very useful event that shows if there are problems with the deletion of records.
Insert failed	This event is if an insertion into the database is unsuccessful .
Update failed	This event is triggered if an update of the database is unsuccessful .

Special Considerations

- Please refer to the special parser functions associated with the database component. (See page 462)

Creating a New Database or Table

Generally mindmap is used in conjunction with existing data sources. In this case you employ the appropriate driver to supply access to the data. In some cases though, a data source does not exist. mindmap permits you to create your own data source while remaining inside mindmap. Please note that not all types of databases can be created. mindmap will only allow you to create data sources which do not require a special database engine, such as dBase, Excel, or text files.

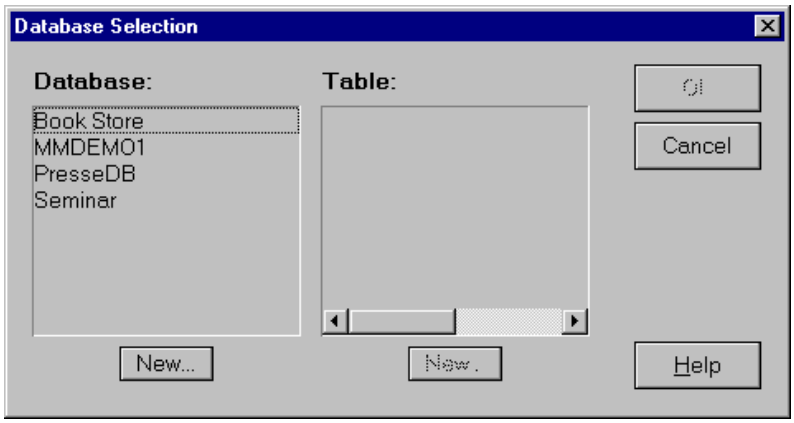
Such files only offer limited database facilities. Many features found in true databases based on the relational model such as Oracle, Informix, Sybase, SQLServer cannot be accessed unless you have the corresponding engine installed on your system.

If you now wish to create a data source, begin by clicking on the icon on the toolbox.

mindmap Toolbox

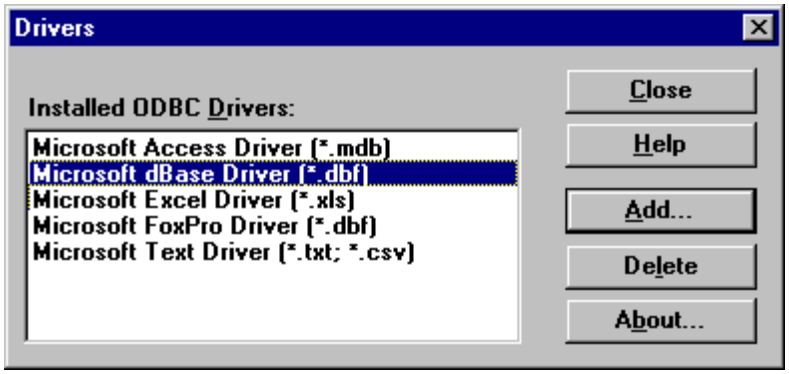
This will cause mindmap to look for all registered database drivers and present you with a list containing the data sources.

If you opted to install the samples during the installation of mindmap, you should have a screen similar to this one. If you are using ODBC in conjunction with other applications, you might have additional data sources installed. If, on the other hand, you did not install ODBC (either with mindmap or with any other installation), then you will have to do so now. Please check out the section **Installation Steps** on page 18 in this manual or refer to your general source of information for your Windows installation.



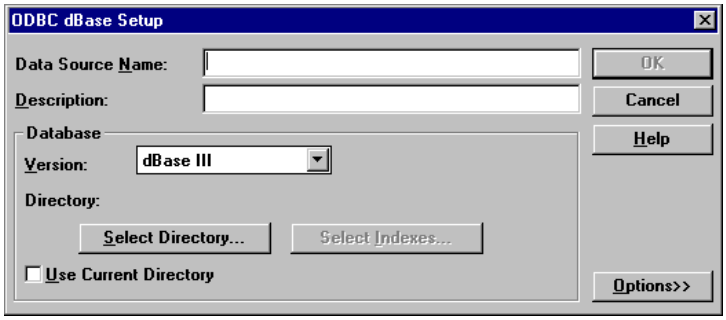
Select from a list of data sources

Since you want to create a new source, click on the left button labeled New on this screen. This will result in the following dialog box, which requires you to select the type of data source you wish to install. Again, please note that the exact appearance of this list depends on your actual installation.



You must specify a database driver to register a new data source for an existing database

For this example, select the dBase driver. This will cause the next screen to be displayed.



This dialog box configures a dBase data source

This screen now offers a number of selections, which are of importance. The first entry is labeled Data Source Name. This field expects a name by which you intend to reference the data source from within mindmap (as well as from any other ODBC compatible applications). Please take care to enter only valid names. The limitations here are:

- a maximum of 32 characters
- must not contain any of the characters
  `[ ] { } < > , ; ? * = ! @ \`

The dialog described here varies from one database system to another. Please be aware that these explanations only apply to the dBase database driver.

Due to a problem in the ODBC driver for dBase, you must first check and then clear the option *Show Deleted Rows*. This will successfully configure the data source so that deleted rows are hidden from the result sets created with this driver.

These restrictions are not imposed or enforced by mindmap. Creating new data sources is dealt with by the ODBC Administrator itself. mindmap merely launches this program. It then deals with the result in the form of data sources.

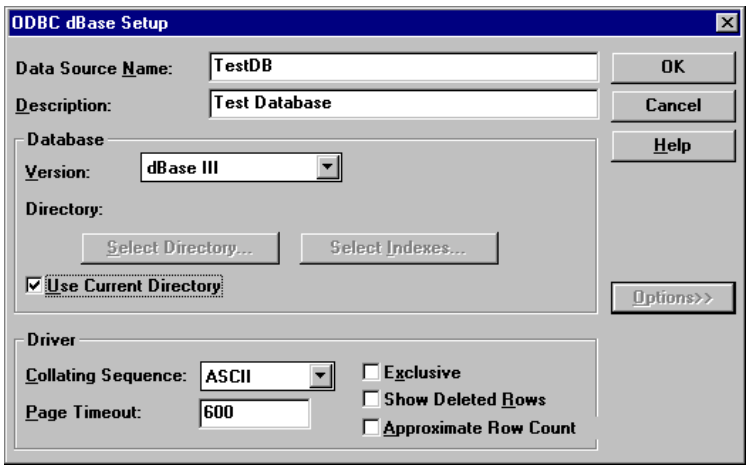
The next field permits you to enter a description for the new data source. Here you can use more than 8 characters and also include blanks.

Assuming that you have selected the dBase file type, the drop down list labeled Version will display this format. In this example, we will use the dBase III file type.

The next selection you must make is the location of the data source. You can either select the box, which will create or locate the new data source in the current directory, or you can navigate to another directory. In this case, clear the check box and click on the button labeled Select Directory. This will pop up the common file list dialog. Make the necessary selections and you will then return to the data source dialog.

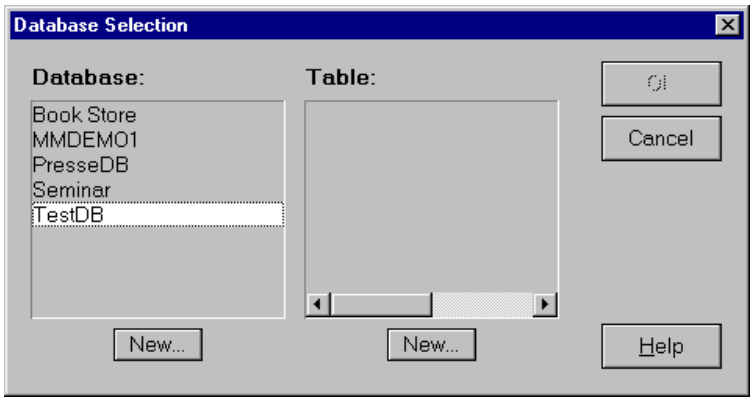
If you are creating this data source for dBase files that already exist, you may optionally Select Indexes to be associated with the dBase tables in the previously specified directory.

Pressing the Options button will extend the dialog box to show more configuration options.



More options to configure a dBase data source

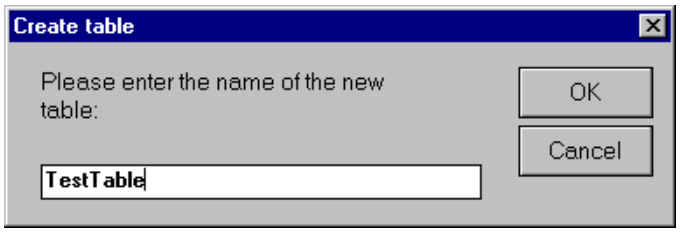
Once you have made the selections, click on the OK button. This will now take you back to the previous dialog box. Now the newly created data source will be listed in the collection of data sources.



The newly registered data source appears in the database list

The next step is to define one or more tables in this database. Please note here, that in the case of non-engine based data sources, you will actually be creating files, in the selected directory. In the case of engine based data sources, the table are a structure contained inside the database management system.

To create a new table, click on the button labeled New… on the right side of the screen. This will display the following dialog box.

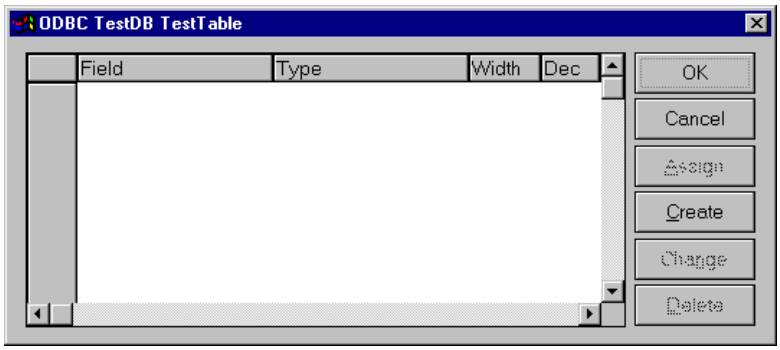


Enter the name of a new table to be created

Enter a name for the table. Make sure that the first eight characters of the name are unique, since ODBC will create a file conforming to the standard MS-DOS file name conventions. In

the case of this example, the resulting file name will be TESTTABL.DBF. Since each database driver maintains a special list of illegal characters, mindmap will attempt to verify the name against this list to make sure that the appropriate table can be created successfully.

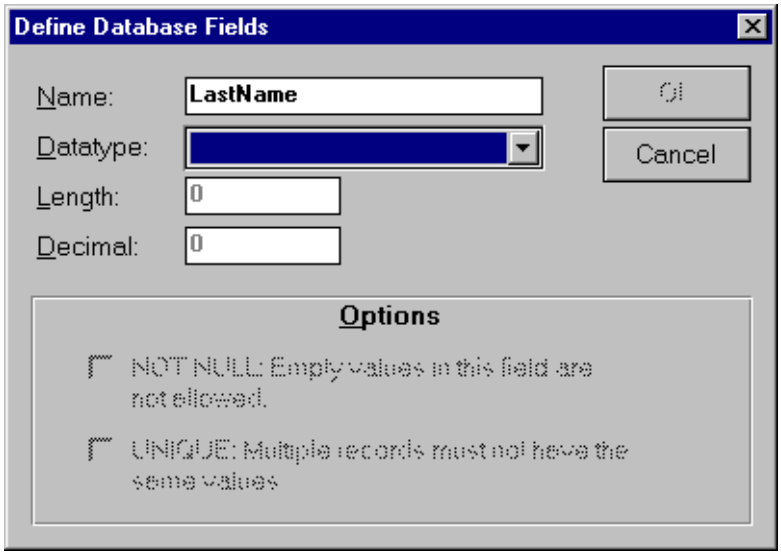
Click on the OK button and the following dialog box will be displayed. It is the dialog box in which you actually define the structure of your table. Initially, the table is obviously empty.



The new table is empty, waiting for your field definitions

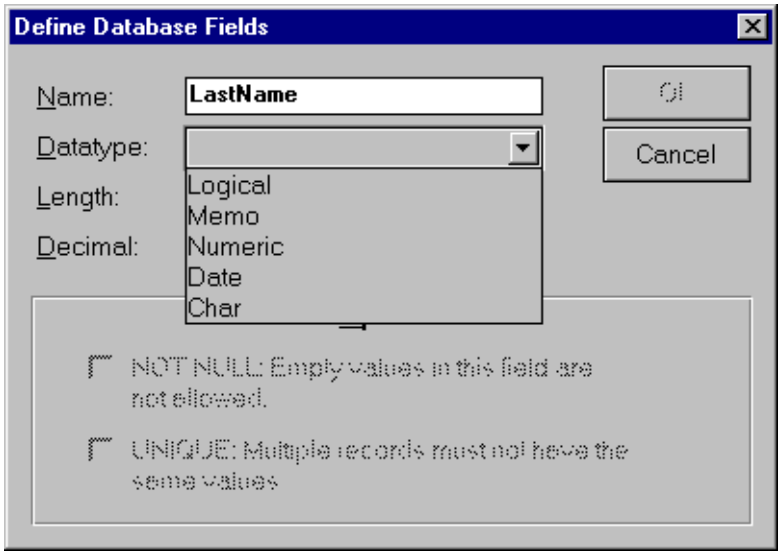
Therefore, you must create a field. Do this by clicking on the Create button on the right side of the screen. You are expected to make four entries for each field you define. Some of the entries are determined based on the data type you select.

We will use addresses as the basis for this little example. Begin by entering the name of the first field contained in the data source; in this case the last name.



Define a new database field and configure various parameters

Again, please note that some databases do not permit the use of names which contain special characters. Some also restrict the length of field names to eight characters. In other cases, some names are reserved for internal usage, such as Date, Char, etc. mindmap will always try to make sure that the names that you enter are valid for the specified database system.

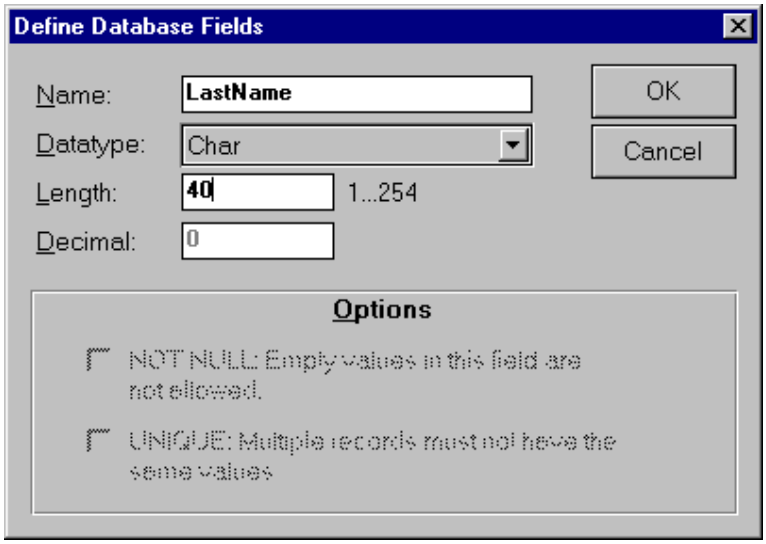


Select from a list of available data types

The next entry required is the type of data the field is to contain. In this case, dBase can distinguish five different types of data:

- Logical – true or false / 0 or 1 / T or F
- Memo – a field which can contain up to 64,000 characters, but which cannot be queried
- Numeric – used to contain numeric data; computations and numerical queries can be performed
- Date – can contain a date (Windows settings are used)
- Char – the most commonly used field type; limited to a maximum of 254 characters

Depending on the selection you make for the field type, you will be offered an appropriate range for the field length.



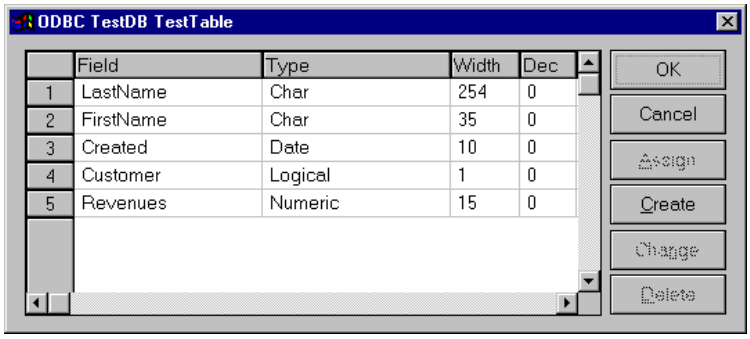
Specify an optional column width

Since the dBase driver does not support special options, the Options section is disabled.

In our little example we have selected a character field, so that the maximum length is 254 characters. Since the field is not numeric, the specification of decimal positions is not valid.

If supported by the database driver, the Options section allows you to control the type of data that can be stored in this database column. Checking the NOT NULL option will prevent empty data from being inserted in this column. By checking the UNIQUE option, the database system will not allow values to be inserted which have already been stored in another row in this table.

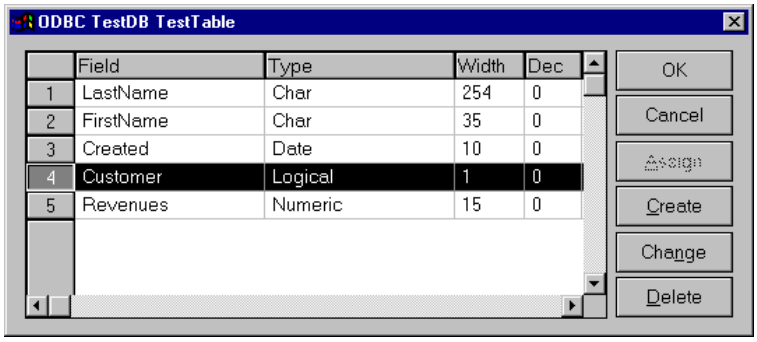
Proceed to define other fields you might want to use in an address application. Once you have entered a number of fields, your table will look similar to this structure.



Multiple database fields have been defined

Please be aware that most database drivers do not allow database fields to be changed after the table has been created successfully.

If you decide to change some of the settings for a field, select the field in this table. The complete row will be highlighted and the button labeled Change will become active. Click on it to make the desired changes.



Highlight a row and reconfigure the settings for this database field

Once you have entered all the fields you wish to use in your little application, click on the OK button. mindmap will now automatically launch the database wizard.

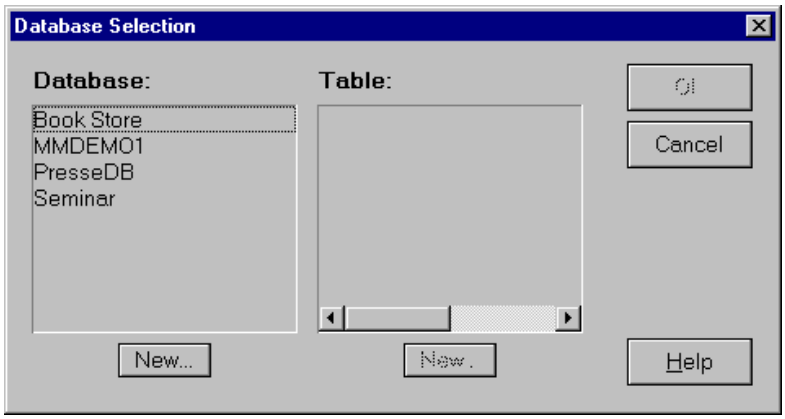
You can create new fields later, so go ahead and click on the OK button, now. Please note again, that the ability to create, delete or change field names is determined by the database or its driver. It is not a limitation imposed by mindmap.

Using the wizard for an Existing Database

mindmap Toolbox

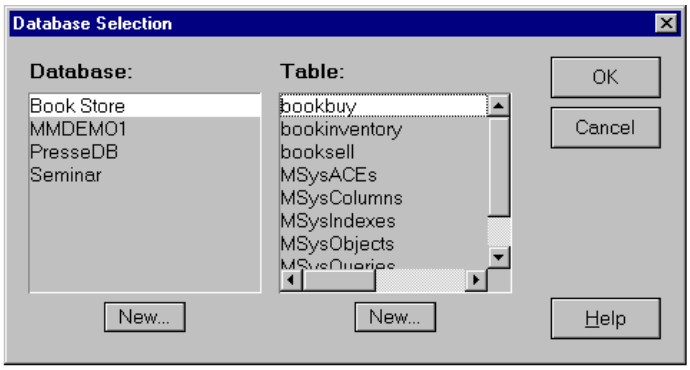
To use the mindmap database wizard, click on the icon on the toolbox.

This will display a list of existing data source.



First, select the database with which you want to work

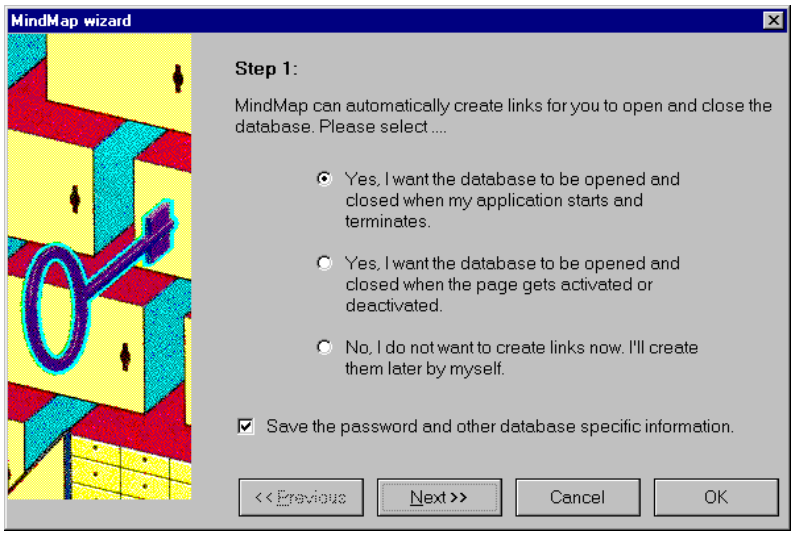
Begin by selecting the database. This will result in mindmap displaying all tables defined in association with this database.



Select from a list of available database tables

This list will contain all tables contained in the database, including those which are reserved for database management functions. You should not select one of these tables unless you know exactly what you are doing.

Once you have selected the table you wish to work with, click on the OK button. The cursor in mindmap will change to a cross-hair with the data source icon attached. Find an empty space on the screen and click the left mouse button once. This will trigger the database wizard and you will see the following screen.



The first wizard page lets you select how to open the database

Three options are offered:

- Open the database whenever the application is opened and close it once the application is terminated (default setting).
- Open the database as soon as the page it has been placed on is activated and close it as soon as the page is deactivated.
- Don't let the wizard create the links.

You can reduce the load time of an application which uses databases by not checking fields every time you open the database. Please see Database component for more details.

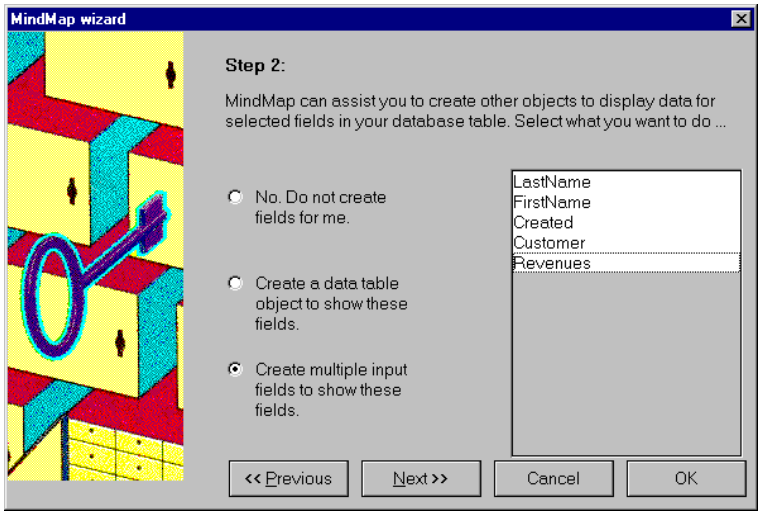
The default setting is the one most commonly used. It will assure that the database is open prior to any activities associated with the database. Depending on the size of the database, it might appear that it takes a long time for the application to begin. We recommend initially using this setting though.

The second option would have the database open whenever you activate the page on which it has been placed. The advantage to this strategy is that the load time for the application is seemingly reduced, since the database is not opened at the start of the application. Activating the page will cause the database to be opened. There are two potential disadvantages to this method. Depending on the size of the database, it might seem to take fairly long to activate the page. You also run the risk of attempting to perform database operations prior to it being opened. This strategy is only suggested if you intend on using many databases and the one you are placing at the moment is small in size and limited in its use during the course of the application.

The third option will prevent the wizard from automatically creating the links. Select this option only if you know how to use databases in mindmap. Opening and closing databases can become tricky, so it is generally suggested that you use the default settings, at least until you have a fundamental understanding of database operations in mindmap.

At the bottom of the dialog box you can also determine whether you want to store database specific information in the link opening the database. Each database maintains a set of configuration options. mindmap will store this information as a string, if this option is checked. We strongly recommend that you examine the link created by the wizard to assure that no sensitive information (such as a password) is contained in this string. Some database drivers even save the exact location of the database file. Therefore, if you intend to ship your application using mindmap's deployment feature, you must make sure that the database will be installed in the appropriate directory on the systems to which it is deployed.

To continue using the wizard, click on the Next>> button. This will take you to the next screen:



The second wizard page lets you select various database fields that can be used in your application

This dialog box offers the fields that are contained in this table. You have the option of:

- not displaying any fields
- showing one or more fields in a data table
- showing one or more fields in individual input fields.

Select the first option, if the table is to be used for purposes other than displaying the contents of fields.

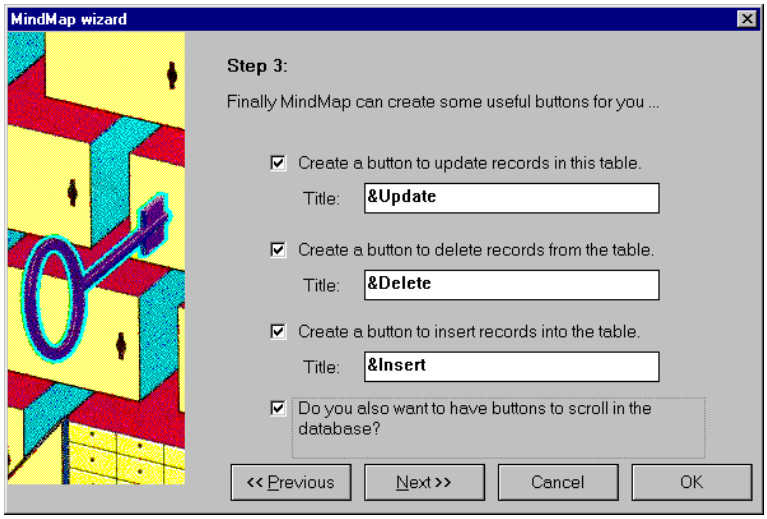
The second option will display the selected fields in a data table. Select the fields you wish to have displayed at run time. You can select one field by clicking on it. If you want to select multiple fields, either press the **Ctrl**-key and click on the desired fields, or press the **Shift**-key and click on the fields. You can also hold the left mouse pressed and drag the mouse over all fields you wish to have displayed. The **Ctrl**-key alternative allows you to select non-adjacent fields in the list.

The column headers will be set to the field names. The order within the data table will correspond to the order displayed in the list. If you wish to change the column headers or the other settings, use the component specific attribute option of the data table to make the desired changes after it has been created.

For both data tables and input fields, mindmap will adjust the formatting options to reflect the data types of the database appropriately.

The third option will create an input field for each field selection. It will also place the field name as a label to the immediate left of each input field it creates. The wizard will begin placing the label/input field combination in the top left corner of the mindmap page on which the database component has been placed. It will then create a column of label/input fields until it reaches the bottom of the page. It will then create a second column and fill until it detects the bottom of the page again. This will continue until either all input fields have been placed or the bottom right hand corner of the mindmap page has been reached. If you have selected more input fields than mindmap can automatically place on the page, it will truncate the list. You will have to either manually size components placed on the screen and then separately place the truncated fields, or you will have to place them on a different page. You still have the option of using a scrollable data table, also.

Once you have selected the appropriate option and the fields you wish to have displayed, click on the Next>> button. If you have chosen not to display any fields, the Next>> button will not be active and you can exit the wizard by clicking on the OK button. In case you do click on the Next>> button, the following dialog box will be presented.

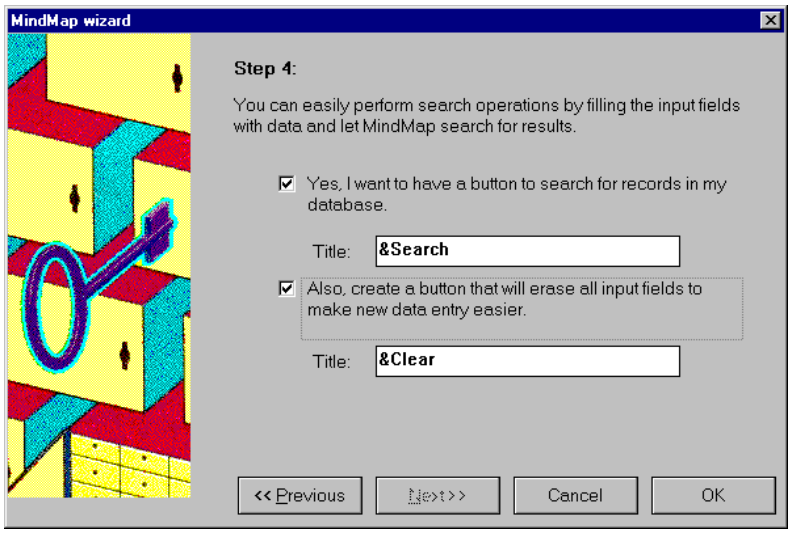


The third wizard page will cause *mindmap* to automatically create buttons to control the database component

This screen will automatically create various buttons and the associated links for you. Select the corresponding check box to have mindmap create the button for you. The default title for each button is also displayed, but feel free to change it. The leading ampersand denotes that the next letter will be used as the accelerator key for the button. You may place the ampersand anywhere in the title, but you must maintain the uniqueness of the selected character. These fields are not parsed, so you may not specify a component name in these fields. If you wish to keep the titles of the buttons variable, wait until the buttons have been created by the wizard and then make the desired settings in the component specific attribute for the button.

The wizard will also create scroll buttons, if you check the last option.

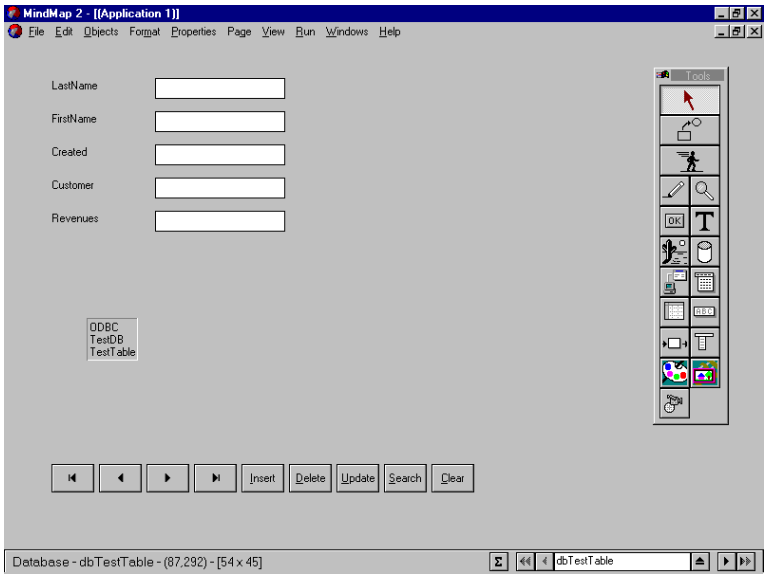
Again, accept the selections by clicking on the Next>> button. This will take you to the last screen presented by the wizard.



If you want to specify search criteria for your database, fill out the fourth wizard page

Select the two boxes if you wish both buttons to be created automatically.

Exit the wizard by clicking on the OK button. This will result in the screen being created, as shown in this screen shot.

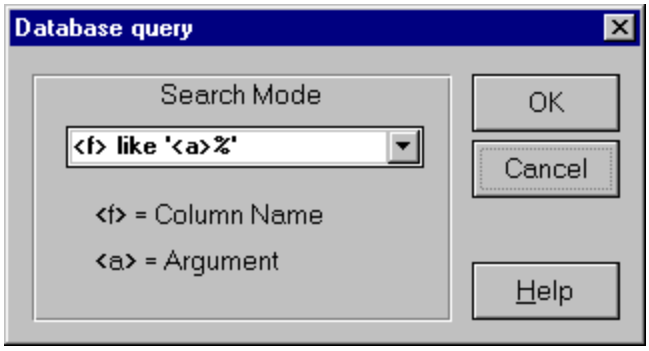


Your *mindmap* application will look like this, after the database wizard has created the selected input fields and buttons

Feel free to make any modifications you deem appropriate. You might also want to look at the links the wizard has generated on the database component and on the buttons. This will further help you in becoming familiar with the underlying logic and processes used in building a mindmap application.

Special Considerations

The default query setting for each input field generated by the database wizard is displayed in the following dialog box.



This dialog lets you define how an input field affects queries to a database

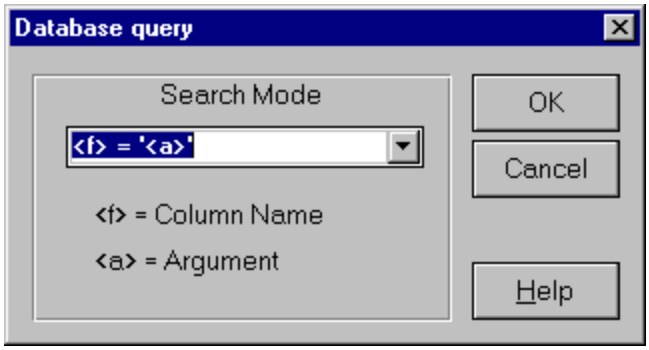
Most database systems do not allow you to query columns that can contain a larger amount of data (e.g. BINARY, LONG VARCHAR, etc.). Please refer to the documentation of the driver or the database system for more information.

This will result in a hit if the contents of the connected database field begin with the string contained in the input field.

The other two options are:

- that the contents of the database must exactly match the contents of the input field, or
- that a substring comparison is made.

For non-character data types (for example, numbers, dates, Booleans, etc.), only the `match exactly` option is applicable.

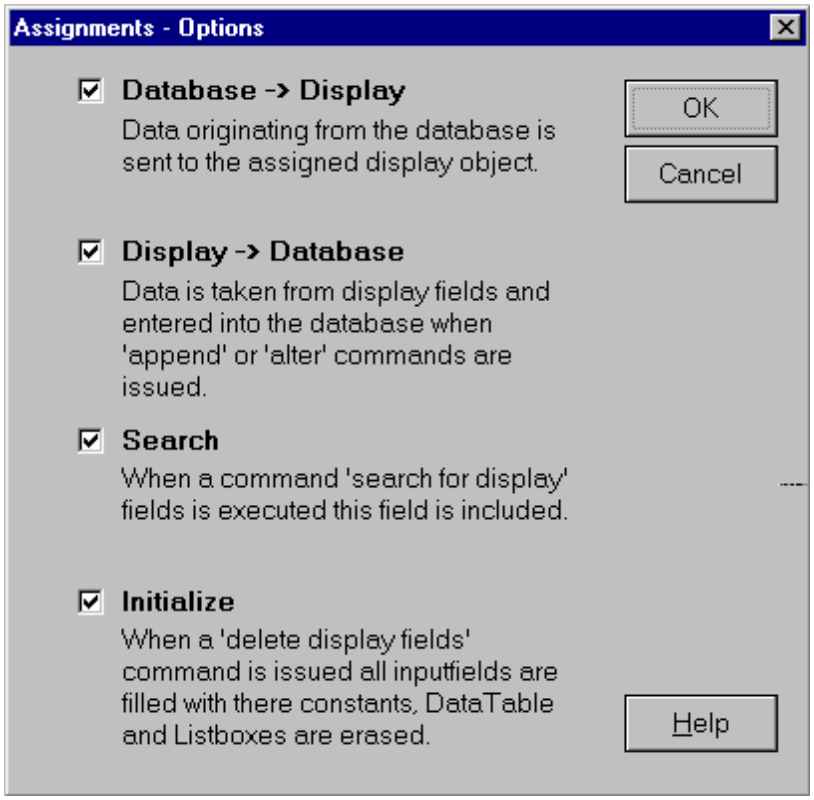


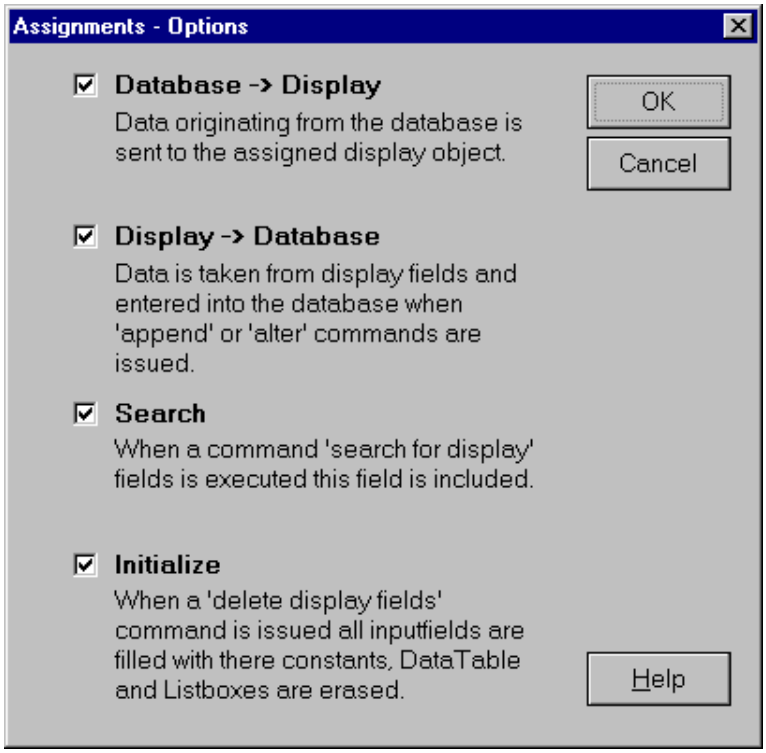
Search option applicable for data types other than string

Binary data types, which can be used to store bitmaps or other raw data, cannot be queried at all.

➤ Find more information about selecting query options for input fields on page 196.

Whenever the wizard creates them, all database fields which are to be displayed -- either in input fields or in a data table-- will have all options selected. However, the Search option is set with respect to the associated data type. mindmap will clear the Search option for LONG BINARY and LONG VARCHAR data types. If your database driver supports queries on these data types, please select this option again.





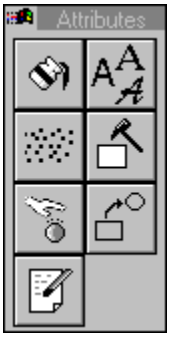
The default settings for visible fields are all selected

If you wish to change any of these settings, access the attribute of the component after it has been created.

List Box/Combo Box Component

List boxes are one of the most often used components in Windows. They are essential for selecting or displaying entries in a list. mindmap offers several types of list boxes that each differ in appearance and functionality. The desired type can be determined in the component specific attributes.

Attributes

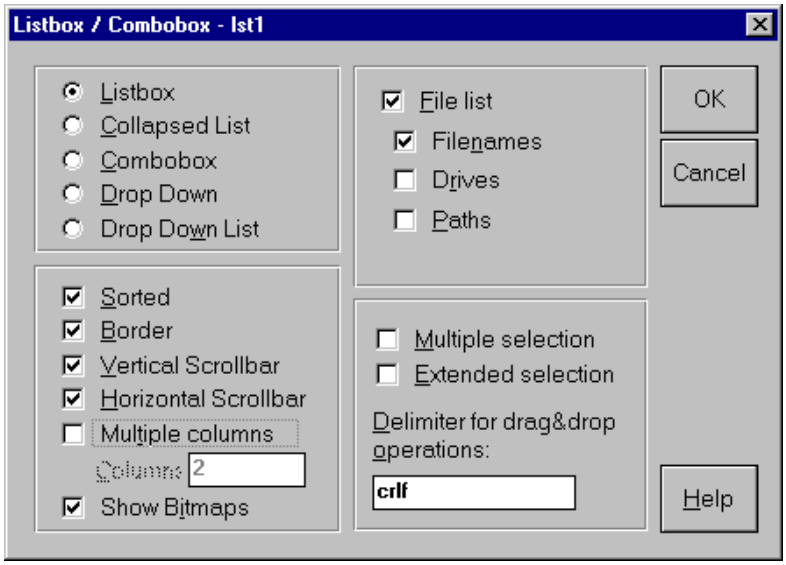


Attributes of list box/ combo box

The Color attribute determines the background color of the list box. The Font attribute permits you to select the font and the color of the font.

† The other common attributes of this component are described in the section Format Menu on page 78.

Component Specific Attribute



Component specific attributes of list boxes

Classes of List Boxes

There are a total of 5 different types of list-/combo boxes. These are:

- List box
- Collapsed list box
- Combo box

- Drop down
- Drop down list

In general, these various types of the same component differ in how they display data and how they accept input from the user. A list box displays data at a single level and does not accept any input from the keyboard. You can drag&drop data to and from it, though. A collapsed list box is the same, with one exception -- it can display data at multiple levels. If the data has a hierarchical structure to it, similar to folders and files in a file system, this type of list box can reflect this structure. It does so by prefixing a row with a ´+´ to symbolize a level containing lower levels. A combo box can be considered the same as a list box with an input field tagged to the top of it. This permits keyboard entry. A drop down only shows the current selected entry of the component. All other entries are hidden behind an arrow icon to the immediate right of the display field. Once the arrow icon is pressed, the list literally drops down to show additional entries. Finally, the drop down list differs from a drop down, in that it will not permit keyboard entry.

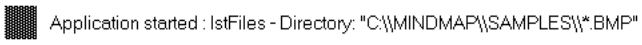
List Box

Entries in a list box

The figure displays the standard list box. Its default setting specifies a visible border. If more entries exist than the list box can display with the selected Font, a vertical scroll bar will be automatically generated. It will only be generated if the check box Vertical Scroll Bar has been selected in the component specific attributes. mindmap automatically defaults to this option. If you desire, you can also have a horizontal scroll bar generated.

If you wish to have the entries displayed in sorted order, then you must make the appropriate selection in the component specific attribute. The list box also offers the option of displaying values in multiple columns. Again, the number of columns is set in the component specific attribute dialog box.

You can also specify that the list box is to be used to display files. The component specific attribute settings permit you to select whether you wish to have only file names, drives, or paths displayed. If you select this attribute, mindmap offers a dialog in which you can specify which files you would like to have displayed. The result of this dialog is a link:



The standard list box has two component specific attributes which relate to multiple selection. If you check the Multiple Selection, then a selection of an entry in the list box will remain highlighted, even when you make another selection. In order to deselect an entry, you must re-select it. The option Extended Selection allows the selection by also pressing the **SHIFT** and/or **CTRL** keys, to specify a range of entries.

Pressing the SHIFT key and clicking on an entry will highlight all entries between the first and the subsequent selection. If you press the CTRL key and make selections, you can highlight non-contiguous ranges.

The Multiple Selection setting also offers the specification of a Delimiter for drag&drop operations. Assume that an input field contains the entry 'John;Martin;James'. If the delimiter has been set to ';' and a drag&drop operation to a list box lst1 is performed, the list box will subsequently contain three entries (rows) -- each entry representing one of the names. If on the other hand, a delimiter has not been defined, then the same drag&drop operation will result in one entry (row) containing the string 'John;Martin;James'.

➤ More information about drag&drop can be found on page 261.

An additional setting permits bitmaps to be displayed in the list box. Place a bitmap on the screen and set its attributes to allow drag&drop operations. Next, select the Show bitmaps option on the specific attributes of the list box. Finally, you must also drag&drop enable the list box. Go into run mode. Click on the bitmap, keep the Left mouse button pressed and move it into the list box. Once you let go of the left mouse button, the bitmap will be sized and displayed in the list box. The list box in the menu option **File | Preferences** offers an example of what this might look like in use.

You can fill a list box either by executing a link or by directing a data source into it. Since this field is parsed, you can also declare a variable (such as txt1) and then have the value of the variable be assigned to the list box. The link commands associated with list boxes and combo boxes are described on page 295.

The value of a list box corresponds to the selected entry. If no entry is selected, the value of the list box is an empty string.

This behavior can lead to the following situation. If you have set the list box to Multiple Selection and you have deselected an entry which doesn't happen to be the first entry, and no

other entries have been selected, the list box will have the last deselected entry as its value. This is due to the last deselected entry still having the focus.

A list box set to single selection will always display the selected entry as its value, or if nothing has been selected, an empty string. Selecting an entry in a single selection list box is identical to setting the focus on an entry. In a list box with multiple selection, the selection does not necessarily correspond to the focus. The focus is always on the last selection (or clear selection).

If you wish to determine whether the value of the list box corresponds to a selected entry, you can employ the following trick. Drag&Drop the list box to an invisible (presumably) input field. If the length of the input field is greater than zero, the entry in the list box had the focus and was selected.

Collapsed List

Entries in a collapsed list

This is a hierarchical list. The preceding ´+´ sign, symbolizes that entries beneath this entry exist. If you double-click on such an entry, the list will expand to show the other entries. These entries are slightly indented to make the list easier to read. Entries on this level can again represent another level. If this is the case, these entries have a preceding ´+´ sign themselves. In the above example, the entry 'New York' belongs to the third level. The top level is 'Western Hemisphere', the second level containing 'New York' is entitled ´USA´. There are no practical limits to the depth of the hierarchy.

The component specific attributes permit the selection of a border and/or vertical and horizontal scroll bars for collapsed list boxes. The options File list, Sorted, Multiple Columns, Show Bitmaps, and Multiple/ Extended Selection have no meaning for collapsed list boxes.

Please take note that the commands Collapse List and Expand List in the message section associated with List-/Combo boxes are only available for this special type of list box.

The collapsed list must always be filled from an external data source or the list box specific link commands Add string or Add string unique. Entries in the data source which do not begin with a ´\´ will be displayed as top level entries. If an entry in the data source is preceded by a ´\´, it will be demoted to the second level and the entry preceding this entry in the data source will receive a ´+´ sign, to symbolize the parent status. If an entry in the data source contains ´\\´, it will be displayed at the third level. The records in the data source corresponding to the above example would have to look like this:

Leading back slashes cause indentations in a collapsed list

The sequence of the entries is crucial for the order in this type of list box!

Entry in Data Source	Displayed in List box
Top Level-A	Top Level-A
Top Level-B	Top Level-B
\SecondLevel-B1	SecondLevel-B1
\\ThirdLevel-B11	ThirdLevel-B11
\\ThirdLevel-B12	ThirdLevel-B12
\SecondLevel-B2	SecondLevel-B2

Combo Box

Entries in a combo box

A combo box corresponds in its appearance to a combination of a list box and an input field. If you make a selection in the 'list box' part, the selection is displayed in the 'input field' portion. Both portions are always visible. The value of the combo box is whatever is displayed in the 'input field' portion.

If the combo box has more entries than can be displayed, mindmap generates a vertical scroll bar. By entering the first characters of an entry, the visible part of the combo box is automatically scrolled to show entries corresponding to the character sequence. This behavior is only displayed if the combo box contains entries which correspond to the character sequence which has been input.

The component specific attribute permits the setting of a border, a vertical scroll bar and whether the entries are to be displayed in a sorted order. The options Horizontal Scroll Bar, Multiple Columns, Show Bitmaps and Multiple/ Extended Selection have no meaning.

You can fill a combo box either via a link or from an external data source.

➤ Find more about list box specific messages on page 295.

Drop Down/ Drop Down List

Entries in a drop down list

These two types of list boxes differ only slightly. Both are displayed as an 'input field' with a button to the right of it. Their behavior only differs in regards to user entry. A drop down allows the user to enter a character string in the upper part of the drop down. It then will position to a corresponding entry, given that one exists. A drop down list does not permit any user entry. If you press a key, the list box will scroll to the first entry corresponding to the character you have entered. This entry will be displayed. To make a selection the user must drop down the list and select from the displayed entries.

Both types of list boxes have the value corresponding to the value displayed in the upper portion of the list box.

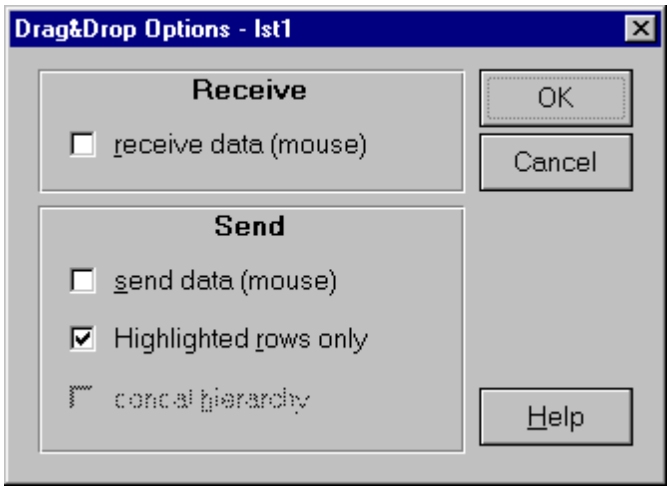
In regards to the component specific attributes, the only difference between both types is that the drop down is able to display bitmaps.

Both drop down and drop down lists can either be filled via links or from an external data source.

➤ See the section Database Manager Component described on page 146.

Attribute Drag&Drop

The drag&drop attribute for collapsed list boxes offers a special option. This allows either only the child or the child and (all) the parent(s) to be returned as the value for the list box. In the above example, not having selected the option concatenate hierarchy would return 'New York' as the value. If the option were selected, the value would be 'Western Hemisphere | USA | New York'.

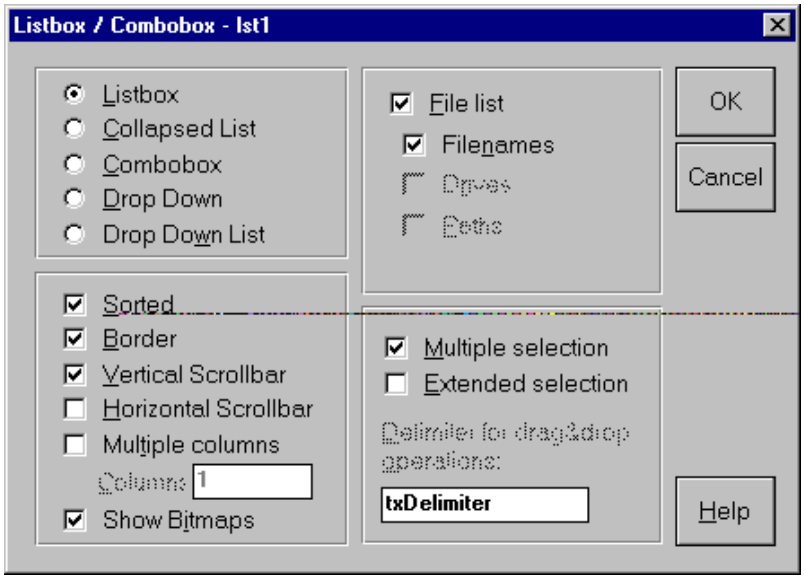


Set drag&drop options specific to a list box

If you want to drag&drop a value from list/ combo boxes, you must select the appropriate entry unless you clear the *Highlighted rows only* check box.

If you wish to drag&drop either to or from a multiple selection or extended selection list / combo box, you must specify the Delimiter for drag&drop operations in the Component Specific Attribute setting. The default setting is `crlf` (carriage return-line feed), which causes each entry to be displayed on its own line. If you are sending the contents of a list / combo box to another component, it is expected that the other component can deal with delimiters (which is true for data tables as well as for multiple line input fields).

It is quite common for one and the same list / combo box to send and receive data via drag&drop. In such cases, it might be necessary to use different delimiters depending on the direction of the data flow. In such a case, define a variable -- presumably a text field -- and assign to it the appropriate string. Find more information about drag&drop on page 261.



Use a text component to specify variable delimiters for drag&drop operations

➤ Other common attributes are described in section **Format Menu** on page 78.

Additional Events

The following table shows the additional events in the links dialog box for list boxes.

Event	Description
Cursor changed	Whenever the user clicks on an entry in the list box, this event will be generated.

Double-click	Whenever the user double-clicks on an entry in the list box, this event will be generated.
Receive focus	Receive focus performs a message when the user places the cursor into the list box or tabs to it.
Lose focus	Lose focus causes an event when the user places the cursor out of the list box; i.e., when the user clicks on another component or when the focus is directed to a different component by pressing the **TAB** key.
Edit Change	This event is for combo boxes only. It is generated if you edit the content of the header row of the combo box.
Drop down	Whenever the user opens the Drop Down or Drop Down List, this event is generated.
File select	This option is only available in conjunction with the component specific attribute, File list. When a file is selected, this event is generated.

Special

- Please note that list/combo boxes offer special parser functions, described on page 467.

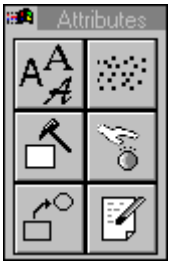
Data Table Component

This component is used to display data in a table format, a bit similar to a spread sheet. The data can either originate from an external data source or can be input via the keyboard. An additional method to populate a data table is by assigning values to it via the parser function SetDataTable (see page 462).

In order to place such a component, click on the icon on the toolbox or select from the **Objects | Data Table** menu. Again the cursor will change its appearance to reflect the selection. It will appear as a crosshair with a little representation of the data table. Move the cursor to the position you wish to place the data table. Then press the left mouse button and drag the mouse to reflect the size you desire to have for the data table. As soon as you release the mouse button, the data table will become visi-

ble. You are then prompted to enter a name for the data table. Either accept the default suggestion or type in a specific name.

Attributes

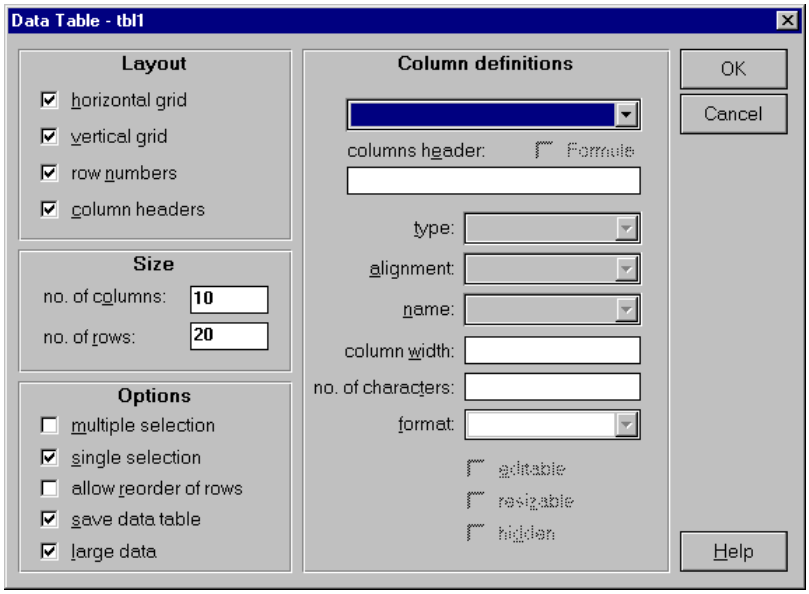


Attributes of a data table component

The Font attribute permits the setting of the common font definitions.

† The common attributes are further explained in the section Format Menu on page 78.

Component Specific Attribute



Component specific attributes of data tables

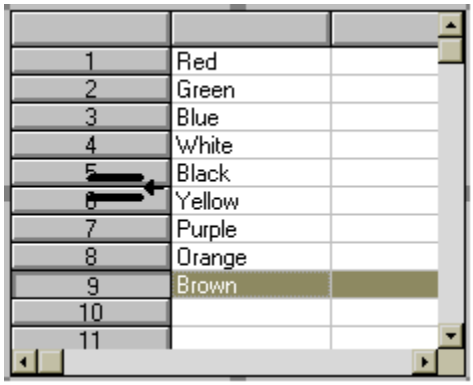
Layout: If necessary, you can decide to have the row numbers display. This might be of assistance when your data table contains many entries. Displaying row numbers is mandatory if you wish to permit rearrangement of rows at runtime. (See the option allow reorder of rows).

The option column headers allows you to enter column headers, given that you have defined such in the column definitions option. You can assign headers that are different from the field's name in the database, assuming that you have connected the data table to a data source.

Size: The layout size function specifies the number of columns and rows. Here you can determine the exact number of rows and/or columns. The selection you make for the number of rows will not be affected at runtime. The setting of the rows will be overridden, if you have connected the data table to a data source and the data set to be displayed contains more rows (= records).

Options: In this section, one may set whether this data table will allow single or multiple selection. The multiple selection highlights every row you click on or a series of consecutive rows if you keep the Left mouse button pressed, whereas the single selection highlights the selected row only (and deselects any other rows previously selected), while every new mouse-click selects another row.

It is sometimes necessary to set the option allow the reorder of rows. This is only possible for data tables with single selection. This option permits the user, at run time, to select a row and move it to another row in the same data table. When performing this operation, moving the cursor above a different row in the data table will make the cursor change to show two horizontal lines with an arrow pointing between them. This is to symbolize inserting the selected one between two existing rows. Once you click on the left mouse button again, assuming the horizontal line cursor is being displayed, it will move the row to the new position. Please remember that it is necessary to have the row numbers display, in order to perform the reordering of them.



A special cursor indicates the ability of moving a row to a different position

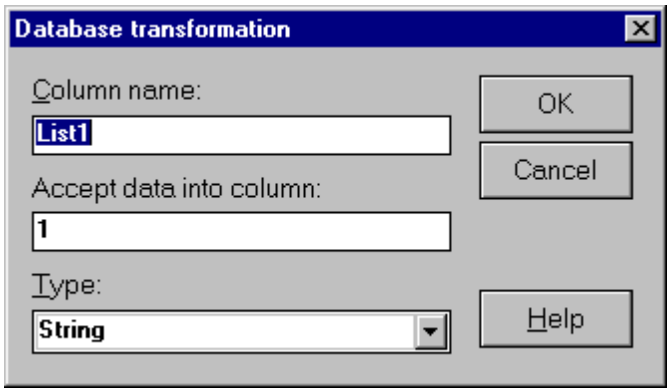
Please note that when you are rearranging rows in a data table, which is connected to a data source, only the row's order in the data table is affected. The order of the records, as they are stored in the external data source, is not affected.

In most cases, the data table is used to display data coming from an external data source. Thus the contents of the data table is constantly changing, depending on the data stream coming in. In some cases though, it is desirable to have a static set of data represented in the data table. In this case, assign desired contents to the data table and select the save data table option. The contents of the data table will then be saved into the application file.

Column definitions: At the top right corner of the dialog box you have the option of defining specific names for each column. Drop down the list to select a column heading. A column is automatically assigned a header consisting of the string 'Column' and an increment. To change it, select the appropriate header and type in the desired name in the field immediately below.

Columns header: Normally you assign a name to a column. If you want to specify the names of the columns, you have to select the appropriate column out of the drop down list that is placed below the column definitions heading. The cursor jumps to the input field below and you can type in the column header. This can be done for all columns. You also can assign a Formula as a column header. This means that the entry will be interpreted by the parser. This allows you to change the column heading dynamically at runtime. This feature can come in handy when developing multilingual applications.

In the case that you are assigning an external data source to the data table, you are prompted, during the assignment, to define a column heading. If you opt to use the default heading, it will be constructed out of the string used as field name in the database. This assignment dialog also permits you to define to which column in the data table you wish to direct the data source. You are also offered an option to define the data type. In all three cases, you can choose to keep the default settings.



Connecting a database field to a data table lets you decide how to configure a particular column

Type: In this section, you select a special type out of the following: String, Float, Integer, Date/Time. The type setting influences the format list. In case the data table is connected to a data source, the data type of the column in the data source is automatically converted to a reasonable type out of the four available data table types. Please check that this type conversion meets the needs of your application.

Alignment: Each column can be formatted with the alignment function. Use this option to set the alignment of the entries in the data table. You can choose from left, right or center aligned.

Name: The alignment of the heading is independent of the alignment of the actual data entries. The header of the column can be aligned left, right or center.

Column width: This setting refers to the visible width of a column. This setting is independent of the option no. of characters. Another way to define the column width can be set with the mouse. If you position the mouse cursor on a header and move

to the edge of the header the cursor will change to a cross with arrows on the left and right side. Press the left mouse button and pull the column width to the desired size. This new size will be shown in the column definition section, when you access the component specific attribute dialog box the next time.

If you connect a data source to the data table, the database system will automatically change the column width to reflect the widths of the appropriate database columns.

No. of characters: The number of characters for each column can be specified here. mindmap sets the number of characters to 80 by default. Depending on the setting of column width the visible portion could be larger. The column width must not exceed 1024 characters.

Format: Depending on the type of column different formats are possible. The entries in the list are merely meant as examples. You can alter these at anytime.

Column type	Possible Formats	Description
String	@@@@	Four alpha-numerical characters can be entered.
	Aaaaaaaa	Eight alpha-numerical characters or punctuation marks can be entered.
	nnn-nnn-nnnn	Ten alpha-numerical characters or punctuation marks can be entered. mindmap automatically supplies the hyphenations at the specified positions.
Float	#,##0.00	The number is represented with two decimal digits. The thousands are separated with a comma. Example: 34,112.32
	#.##	The number is represented with two decimal digits. Example: 34112.32

	+#.####	The number is preceded with a plus sign and has four decimal digits.
Integer	#.##0.00	If a number with decimal digits is entered, these are truncated. The thousands are separated with a comma. Example: 23,345.00
	#.##	If a number with decimal digits is entered, these are truncated. Example: 3.78 becomes 3.00
	+#.####	A number with four decimal digits will be preceded with a plus sign. Example: +2341.0000
Date/Time	DDD MM/DD/YYYY	The date is displayed with a Day of Week abbreviation. The month is represented as a two-digit number. The day is also represented with two digits, the year will be displayed in four digits. Example: Sun 12/03/1995
	DDDD MM/DD/YYYY	In this case the date will be displayed in the same format as in the example above, with the exception that the weekday is not abbreviated. Example: Sunday 12/03/1995
	MM/DD/YY	In this example the weekday is not displayed at all. Example: 12/03/95
	MM/DD/YY HH:MM	The date again is displayed as described in the above examples. The time is appended. It is displayed military style ('00' through '23' hours) and 2 digit minutes. Example: 12/03/95 21:14

	MM/DD/YY hh:mm P	The date is displayed as in the above example, except that the US style of displaying hours in two cycles of '00' to '12' followed by either AM or PM. Example: 12/03/95 09:14 PM

You can adjust the rows to be set to editable, re-sizable or hidden.

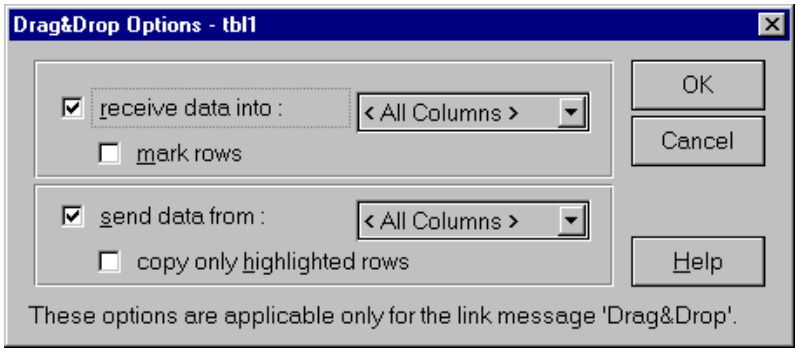
In the case of editable, the user can change the contents of a cell. If you intend to use a data table only to present data (and not allow the user to change the values), you should switch off the option editable.

Re-sizable allows the user to change the column width with the mouse.

You can set the column to hidden so that it is not seen by the user. This feature is very important in conjunction with a data source. You can connect hidden columns in the data table to relevant fields in the data source. Unique keys, numerical sort order indices, etc., can thus be kept synchronized, without having to display them.

Attribute Drag&Drop

The drag&drop dialog box exclusively controls the behavior of the data table when a drag&drop link message is applied. Once the receive data into and/or send data from check boxes are checked, it is possible to move data into and from a single cell, by dragging with the mouse. Please note that dragging information between cells within the same data table is not supported.



Configure various drag&drop options specific to data tables

This dialog box permits you to specify whether the data table is to receive data via a drag&drop link message. You can specify that data is to be received with the receive data into option. If you have made this selection, you can determine whether the data is to be received in all columns (All Columns) or restrict it to individual columns. If you perform the drag&drop link message from a list box or from a multi line input field into a single column of the data table, all rows that match the lines of the drag&drop source will be highlighted. Obviously, this operation only works if the multiple selection check box is checked in the component specific attributes dialog.

If you intend to permit data to be sent, you must select the option send data from. Then specify whether all columns or only the marked columns are to be sent. Copy only highlighted rows allows only selected rows to be sent.

When sending drag&drop data, the data table component will convert its contents in a way that cells within the same row will be separated by the **TAB** character and rows are separated with the CRLF (carriage return/ line feed) character sequence.

If drag&drop data is sent to the data table the various cells will be filled with the information from the drag&drop source component if they are formatted in the same way.

Again, please be aware of the fact that selecting a column for sending or receiving drag&drop information only applies to the drag&drop link message. It does not affect the ability to drag&drop information using the mouse.

➤ Find more information about drag&drop on page 261.

Additional Events

The other additional events are shown in the following table.

Event	Description
Cursor changed	Whenever the user clicks on an entry in the list box, this event will be generated.
Double-click	Whenever the user double-clicks on an entry in the list box, this event will be generated.
Receive focus	Receive focus performs a message when the user places the cursor into the list box or tabs to it.
Lose focus	Lose focus causes an event when the user places the cursor out of the list box; i.e., when he clicks on another component or when the focus is directed to a different component by pressing the **TAB** key.
Abort edit mode	If you have selected a cell in the data table and press the **Esc** key, this event is generated.
Begin edit mode	This event is generated when you begin to input data into a cell.
Cell changed	This event is called as soon as a cell has been changed.
Order changed	The change of the data table's order generates this event. This event is only defined for data tables with the Component Specific Attribute allow reorder of rows set.

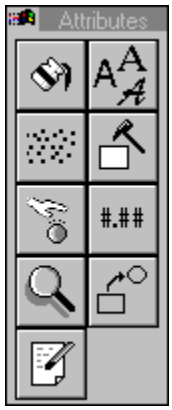
Input Field Component

Input fields are generally used to either get input from the user at runtime or to display data originating somewhere else.

In order to place an input field, click on the icon on the toolbox or make the selection via the menu option. Once you have selected the input field, the cursor will change its appearance and will become a crosshair with a little rectangle attached to symbolize the input field draw mode. Move the mouse to the screen position where you wish to place the input field. Press the left mouse button, drag the mouse towards the lower right screen and release the mouse button. You are immediately prompted for the component name. You can either choose to use the default name or simply type in the new name.

Depending on the font selection, opening an input field over a certain height will produce vertical scroll bars. Vertical scroll bars signal that the input field can display multiple lines.

Attributes

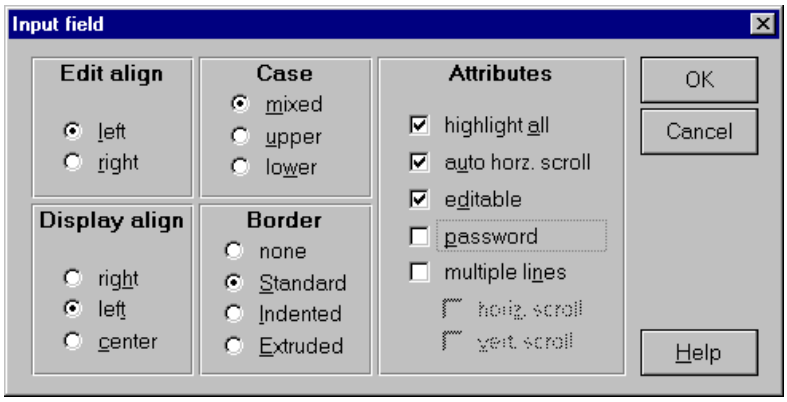


Attributes of the input field component

The Color attribute defines the background color for input fields. The Font icon permits the standard font settings, as well as the color definition for the font.

† For additional information regarding formatting see the section Format Menu on page 78.

Component Specific Attribute



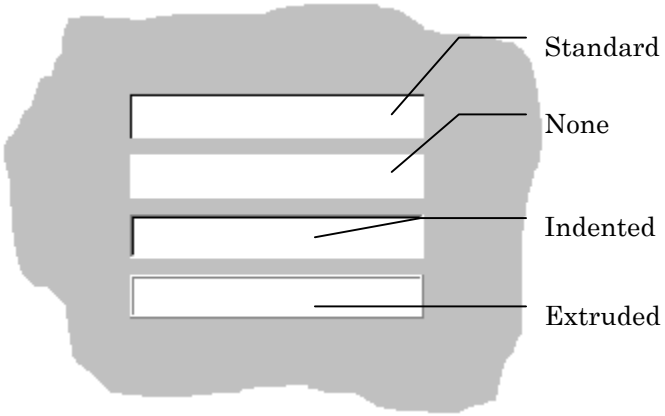
Component specific attributes of input field

The Edit align attribute enables you to define were the text should be aligned when the cursor is placed in the input field. The options are left or right.

Independent of the selection you chose for alignment when editing, the setting of Display align will take effect once the input field loses the focus. The Display align option allows right, left, or center alignment.

The attribute Case permits you to determine how the input field will deal with the user input in regard to capitalization. If you select the setting mixed (default setting), the input will be displayed in exactly the same manner the user inputs it. The other two alternatives are upper and lower. In the case of upper, all input will be displayed in capital letters, regardless of how the user inputs the characters. If you have chosen lower, then all characters will be displayed in lower case, again regardless of how they were input. Please note that this setting also affects how the content of the input field is passed to other components, especially database components.

The Border option allows you to choose between various settings, which merely affect the appearance of the input field. These settings are specific to the input field, so they can be found under the component specific attribute.



Input field with various border settings

The highlight all option will select all characters in the input field, once the input field receives the focus. If the user begins to type, the selected characters are deleted (overwrite mode). If

you do not check this option, then the cursor will be set to the left of the first character in the input field, once the input field receives the focus (insert mode).

Sometimes there may not be enough space for long input fields on your page. If you do have long strings, you can set the Input field to automatic horizontal scroll. This allows you to have a small input field and the user can scroll through the string to view it completely.

You will not have scroll bars with scrolling input fields, unless they are set to have multiple lines. The user must position the cursor in the input field and move to the end to see the whole string.

A better way is to use multiple lines for those input fields with long strings. If you choose this option, the edit align, display align and case section are grayed out and they are not accessible. If you decided to use multiple line input fields, you may select to have horizontal and/or vertical scroll bars.

Input fields can be set to editable or not.

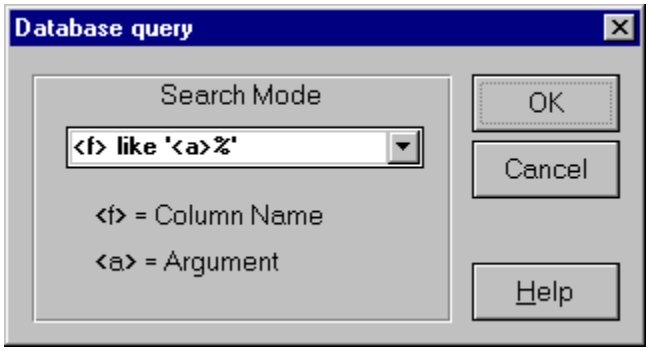
A password function is integrated, as well. This function shows only asterisks in the input field and the user's input is safe and hidden. However, if you assign the value of the input field to another component, the actual password (not the asterisks) will be transferred.

Database Query

You can perform a database search on the content of an Input field that is connected to the database. (See the section **Database Manager Component** on page 146). The search mode you defined here will be used.

These search mode settings are valid for the commands Search for fields. If you dynamically construct your own SQL strings at runtime using the SQL Command, you do not need to consider these settings.

The attributes toolbox of input fields offers an icon for database queries. If you click on the icon, you will be offered the following dialog box:



Specify which criteria are to be applied to database queries

If you open the list, you can select from a predefined list of various Search Modes:

Search Mode	Description
<f> = '<a>'	The first mode searches for a value that equals the argument. If the content of the input field <a> is 'test', then the result set will only consist of strings which match this string exactly. Please note that this option is also applicable for searching other data types than strings, such as numeric values, dates and times and Boolean data.
<f> like '<a>%'	The second mode searches for a string that starts like the argument. This condition is satisfied with such results as 'test', 'tests', 'tested', 'tester', etc. This option is applicable only for string searches.
<f> like '%<a>%'	The last mode searches for a part of a string that is like the argument. Achievable results include 'test', 'tested' and 'pre-tested', etc. Again, this option is applicable only for string searches.

‘<f>‘ refers to column name, where the field name in the data source is meant; ‘<a>‘ is the argument, or, in other words, the contents of the input field.

You may have noticed that, besides selecting one of the predefined values, any text can be entered in the search mode combo box.

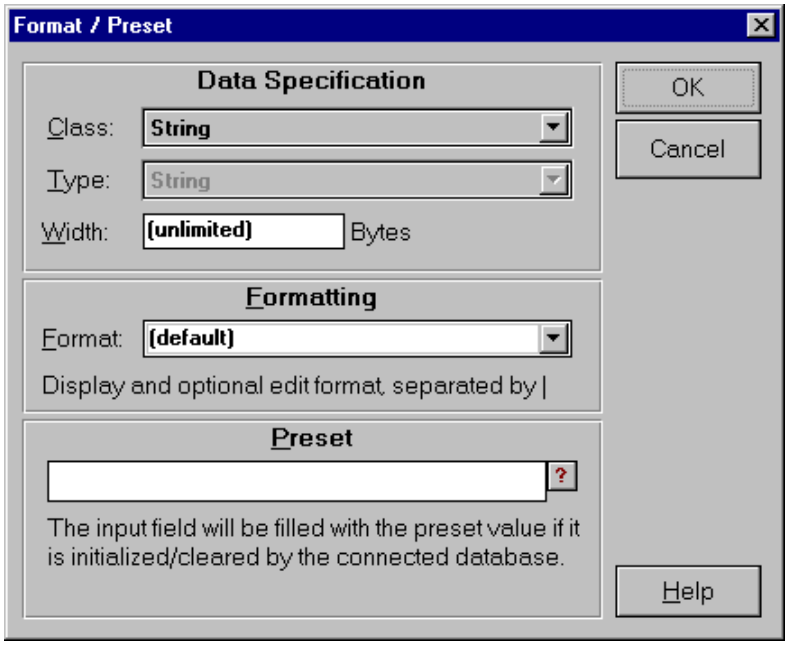
If you notice that a specific database driver supports search conditions other than ‘=’ or ‘like’, you may enter these verbs in this combo box.

For example, try using comparison operators such as <, <=, >, >= and <>.

Also note that the content of this combo box is evaluated by the parser at run time. This means that you are allowed to enter the name of another component, as long as it contains a syntactically correct comparison operator with respect to the database the input field is connected to.

Format/ Preset

Input fields have various formatting options. If you click on this button on the attribute toolbox, you will be presented with the following options:



Control the type of information to be displayed using the input field component

The preset configuration is used in response to the database command *Clear Fields* described on page 308.

If you select the entry Number from the Class list, you will be offered different Types. The Type list contains all data types relating to the selected Class. If only one type is applicable for the selected class, this entry shows the appropriate setting and is disabled.

The Width field allows you to specify how many characters may be entered into the input field - it is only defined for the data type string. The default setting for the Width field in this case is unlimited. Unlimited in this sense means that a limitation depending on available internal system resources applies. Expect this limitation to be around 16000 characters for single line input fields and 32000 characters for multiple line input fields.

If you connect a data source component with an input field, the input field picks up its settings for Class, Type and Width automatically and thus corresponds to the field in the data source.

This conversion is accomplished by using the information that a specific data source returns. Some data sources may supply this information incomplete or not recognizable for the input field. mindmap will attempt to match these types as closely as possible. However, you should always verify, that a reasonable type has been selected.

If you select Format, assuming the data type supports it, you can choose a format from the list or define your own, according to the masks listed below.

Class	Token	Replacement
Boole		
	<verb1>; <verb1>	The first verb will be displayed if the Boolean value is true, whereas the second verb will be displayed if the value is false. Assigning a numerical value of zero to a Boolean input field will set to false, all other values will set to true.
Mask		
	#	Any numeric digit (0-9)
	@	Any alphabetic character (a-z, A-Z)
	!	Any punctuation character
	*	Any single printable character
	\	Causes the next character to be literal
		Any other character will be displayed as is.
Number Currency		If there are two formats separated by ; mindmap will use the second format to display negative values.
	#	Any numeric digit (0-9)
	.	Decimal point delimiter
	,	Separating delimiter for thousands
	+ or -	plus or minus sign
	\	Causes the next character to be literal
		Any other character will be displayed as is.

Date		The following tokens can be concatenated to create a date format using any reasonable delimiting characters as can be seen from the following examples: mm/dd/yy → 08/28/98 mm/dd/yyyy → 08/28/1998 Dddd, Mmm dd, yyyy → Friday, Aug 28, 1998
	m	Shows the month. (1..12)
	mm	Shows the month using two digits with preceding 0. (01..12)
	d	Shows the day. (1..31)
	dd	Shows the day using two digits with preceding 0. (01..31)
	Ddd	Shows the day-of-week abbreviated to 3 digits. (Sun..Sat)
	Dddd	Shows the day-of-week. (Sunday..Saturday)
	Mmm	Shows the name of the month abbreviated to 3 digits. (Jan..Dec)
	Mmmm	Shows the name of the month. (January..December)
	yy	Shows the year using two digits.
	yyyy	Shows the year using four digits (therefore including the century).
Time		The following tokens can be concatenated to create a date format using any reasonable delimiting characters.
	h	Shows the hour. (0..23)
	hh	Shows the hour using two digits with preceding 0. (00..23)
	m	Shows the minute. (0..59)
	mm	Shows the minute using two digits with preceding 0. (00..59)
	s	Shows the second. (0..59)
	ss	Shows the second using two digits with preceding 0. (00..59)

Date/ Time		This format allows a concatenation of the tokens described for Date and Time.

Please note that you may specify two formats like this separated by the | character. If you do so, the second format will be used if the input field has the focus (is in an editable state). This is useful especially for date values containing the day-of-week, since this value depends on the date settings and may not be specified independently. If no second format is given, mindmap attempts to use the same format for both edit and display mode.

The Preset option allows you to preset the contents of an input field, if and only if, it has been connected to a data source. If you issue the database command Clear display fields and have set the option Initialize on, the contents of the field will be cleared. If the Preset option has been selected, the associated value will be displayed. Please note that this value is evaluated by the parser.

➤ See section **Database Manager Component** on page 146 and a description of database related links on page 252.

Additional Events

The create and edit links dialog box for input fields has some of the standard events. The few additional events are shown in the following table:

Event	Description
Enter	This event is generated when the cursor is positioned in an input field and the user pressed the *ENTER*-key.
Keyboard input	This event is generated when the cursor is positioned in the input field and the user presses any key.
Receive focus	This event is generated when the component receives the focus; for example, if you select it.
Lose focus	This event is generated, as soon as the input field loses the focus.

Input / Output Component

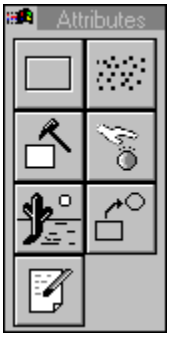
Via the File and Clipboard interface, you can import and export data (drag&drop). The Printer component permits you to output information to the printer, also via drag&drop.

Once you have clicked the input/output symbol or the corresponding menu option, you will see the following dialog box:

Initial dialog box to select the type of input/ output component

Here you can select the appropriate symbol for File, Clipboard or Printer. Once you have selected one of the three options, the dialog box will be updated to reflect the resulting options. The resulting dialog box corresponds directly to the one you would see, had you selected the component specific attributes. Once you have clicked the OK button, the cursor will change its appearance to look like a crosshair with a little attached input/output component symbol. Move the cursor to the desired position on the page and drop the component. The component will be created in its standard size. Obviously, you can also drag open the component (Left mouse button pressed while dragging). In this case, the size of the bitmap will always remain the same size and only the surrounding frame will become larger.

Attributes



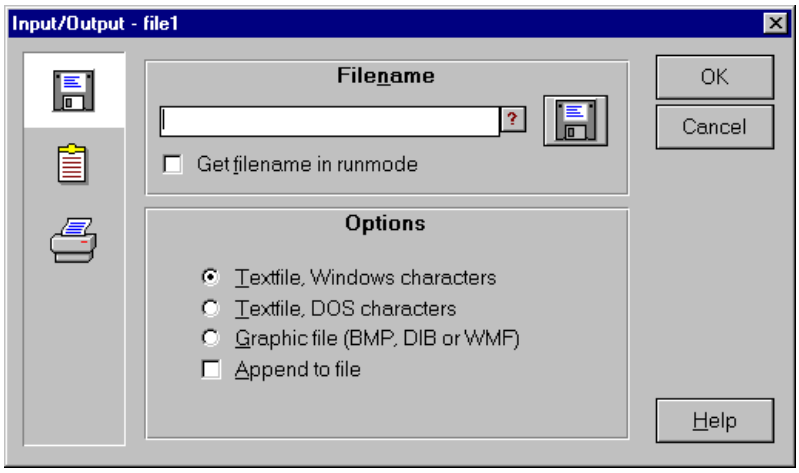
Attributes of Input/ Output component

The default setting for the drag&drop attribute is send data as well as receive data. The attribute Graphic Import offers the option of replacing the bitmap of the placed component with one of your own choice.

† For further information concerning common attributes see the section **Format Menu** on page 78.

File

This component is designed for reading data from or writing data into a file. When you select the symbol, the dialog box will appear as follows:



Configure the settings to work with files

This dialog is identical to the one which appears when you select the component specific attributes.

In this dialog box, you can input a file name with directory and path (don't forget to use two backslashes to identify the directory name) or select a file using the button with the diskette symbol. The entry for the file name is parsed so that you may also enter a component name containing the desired file name.

If you select the option get file name in run mode, you do not need to make an entry for file name. At runtime, a dialog box will appear as soon as the drag&drop link is executed. Here the user can enter a path and file name.

In the Options section, either a Windows or DOS characters text file can be defined. A graphic file (BMP, DIB or WMF) could also be used. The text in the Input field could be appended to the text file by checking the appropriate box.

These components communicate via the drag&drop operation. In principle there are two different ways to execute the drag&drop operation. (See also **Properties | Drag&Drop Options** on page 94 and the description of the link on page 261). Independent of how the drag&drop option has been set, you can include this link. Such a link could look like this:

Left mouse button released: Drag&Drop file1 → edt1

The above link would display the contents of the file specified in file1 in the input field edt1, once the user has pressed the button. If you exchange the direction of the link, then the contents of the input field edt1 would be stored in a file specified in the output component file1.

In order for this to function, it is not necessary to have the file component displayed on the screen.

The other method to execute a drag&drop operation is by means of the mouse. In this case, it is not necessary to create a link. You merely set the attributes of the sending and receiving components so that both can participate in the operation. You must set the receiving component's drag&drop option to receive data. The sending component (or rather the component from which the data is originating) must be set to send data.

To receive the same results as in the above example (display the contents of a text file in the input field), at run time, move the mouse over the file component. Then press the left mouse button and drag the mouse to the input field. Once the cursor is above the input field, release the left mouse button. As a result, the contents of the file will appear in the input field. If you have chosen to get file name in run mode, a dialog box will appear, offering the opportunity to select a file. Once the file has been selected, its contents will appear in the input field.

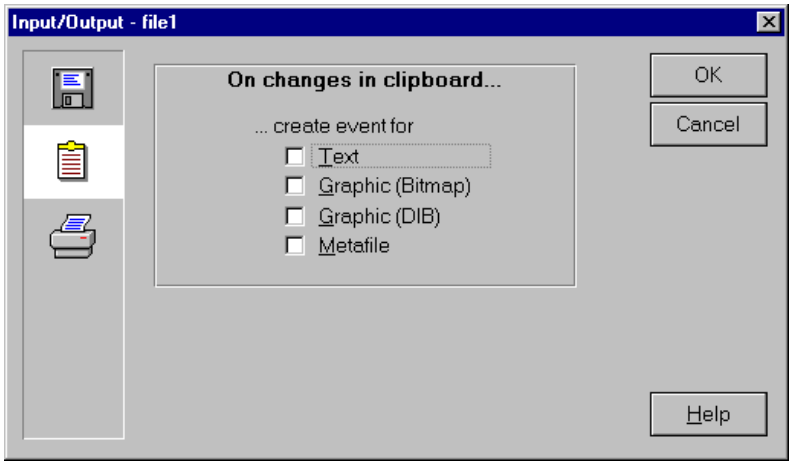
In this case, the file component must be visible on the desktop at run time.

Special Note

mindmap offers the facility of storing the contents of a data table in a file and, vice versa, of displaying the contents of a file in a data table. The data in the columns is separated by a TAB. The beginning of a new row is defined by a CRLF (Carriage return line feed).

Clipboard

The input/output clipboard component is similar to the file component. You can send data from the Windows clipboard to an input field and vice versa. If you select the clipboard symbol, the dialog box will appear as follows:



Dialog box for an input/ output component to control the interaction with the clipboard

On changes in the clipboard, mindmap creates events for text or graphics (BMP, DIB or WMF).

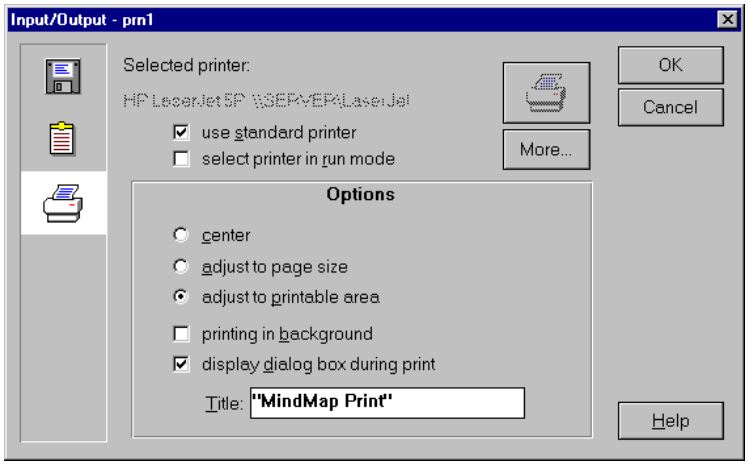
Component Specific Attributes

This has identical functions to the file component.

Event	Description
New data in clipboard	Whenever the Windows clipboard is filled by another application, this event is generated.
Clipboard empty	This event is generated when the clipboard is emptied by another application.

Printer

This component allows for printing in an application. If you select the printer icon, the dialog box will appear as:



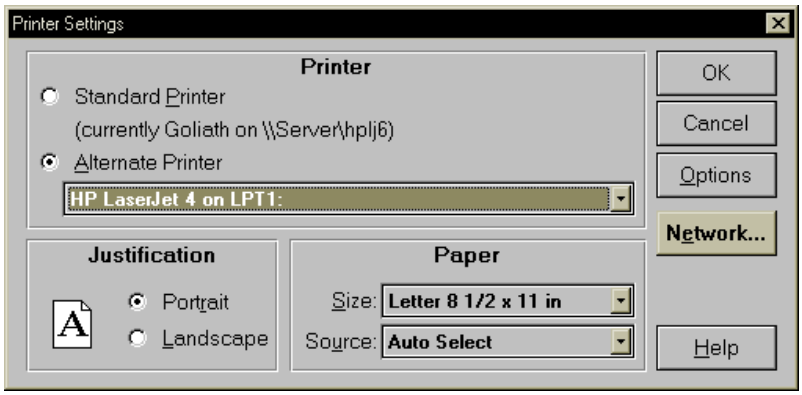
This dialog defines how and to which printer the output is supposed to be sent

In the first section, mindmap shows the selected printer. By default, this is your standard Windows printer -- the standard printer of the system on which you are constructing the application.

If you move such an application to another computer, please make sure that exactly the same printer on the same port is available and functioning.

You can choose whether the installed standard printer should be used for printing, or if the user has to select a printer in run mode. The setting standard printer will allow for printing from the application on any computer, if a standard printer has been defined in the control panel. If you clear both check boxes, mindmap will print to the printer displayed as selected printer regardless of which printer has been selected as default printer.

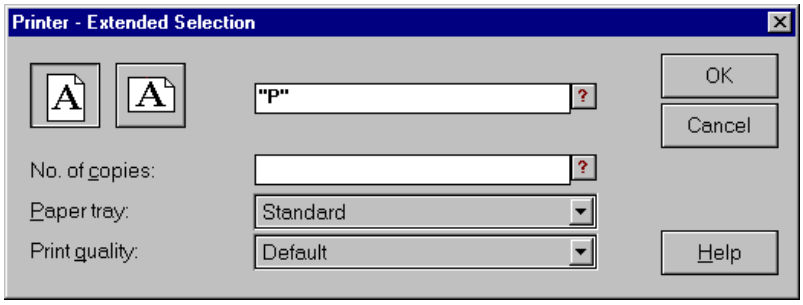
If you click on the button with the printer icon, the printer setup dialog box pops up:



Printer setup dialog

Here you can select a Standard Printer or an Alternate Printer, provided you have installed other printers in Windows (via Control Panel). The Orientation -- portrait or landscape -- can be selected as well (by clicking the button More). The Paper Size and Source can also be changed. These specifications depend on the selected printer and are not described in detail here. The Options button opens your special printer setting where you may specify the print quality, for example.

The More... button opens the Printer - Extended Selection dialog box.



Configure more printer options

The Orientation, Number of copies, the Paper tray and the Print quality can be selected in this dialog box. These settings differ from those in the Printer Setup, in that they are parsed at run time. This allows for the entry of variables (names of components which refer to names).

➤ See page 388 for more information.

The section Options allows additional settings depending on the selected printer.

You can select center, adjust to page size or adjust to printable area, where adjust to page size is the default setting. The center setting will print the text or the graphic, center aligned. If you choose the setting adjust to page size, the printout will always be the same, regardless of the printer selected. Some printers have a wider non-printing margin. This would normally truncate the output. This can be avoided, if you choose the setting adjust to printable area. In this case, you select the printer and, if necessary, the output will be adjusted in size to fit the permissible area.

If you have selected printing in background, you can print in background while continuing to use the application in foreground. You can select the display a dialog box during print option, which will display a dialog box when the printing begins. The Title states the caption for the printer dialog box. The name of your application or the name of the printout can be stated here, for example. Since this field is parsed, you can also include component names which reference text strings.

Special Note

A printer will only permit output. In addition, you have the two options mentioned above, if you wish to initiate an output.

Normally the contents of the output page component is printed. This implies that you place other components (data tables, graphics, etc.) on the output page component. Then you drag&drop the output page to the printer component.

You can also print graphic components directly, without going through the output page component. In this case, you just drag&drop the graphic component directly to the printer component.

➤ For more information see page 261.

Menu Component

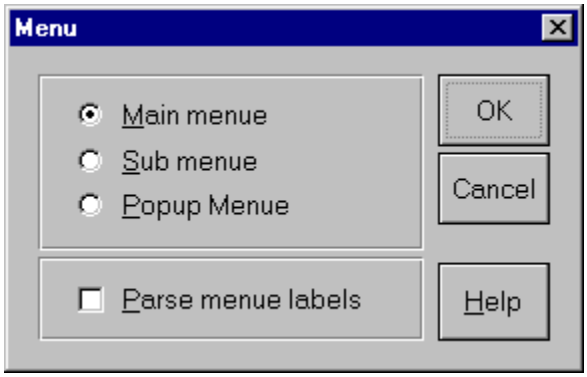
You can create menu items for mindmap applications, much the same as other Windows applications have menus. In order to create menus, click on the menu icon on the toolbox or make the appropriate selection from the menu **Objects | Menu**. The cursor will change its appearance to reflect the selection and will display as a crosshair with an attached menu icon. Move the mouse to the area in which you wish to drop the menu component. Press the left mouse button and drag open the component. A grid will appear and the first row will contain the entry 'Menu'. Placement is not critical on the page, as the component will not be visible at run time. On the other hand, it is critical on which page in the application it is placed. mindmap uses the menus it encounters, until it finds the next menu. Thus, it is important to consider their location, relative to the execution of your application. For example, this allows dynamic menus and context sensitive menus.

Attributes

Attributes of the menu component

† The common attributes of this component are described in the section Format Menu on page 78.

Component Specific Attribute



Component specific attributes of a menu

Once you have placed a menu component, its default setting is Main menu. You do not need to edit this, even if you want it to be a sub menu. The setting will automatically change when two menu components are connected, using the rubber band approach.

The option Parse menu labels is used if you wish to have the contents of the menu entry be dynamic. Normally, you enter a static text which will display at run time. An alternative to this approach is to enter a component name and let the contents of the menu option become the contents of the referenced component. Thus, you can change the menu entry dynamically. In order for mindmap not to take the component name as a literal, you must specify that the menu entries are to be parsed. This is set in the component specific attributes.

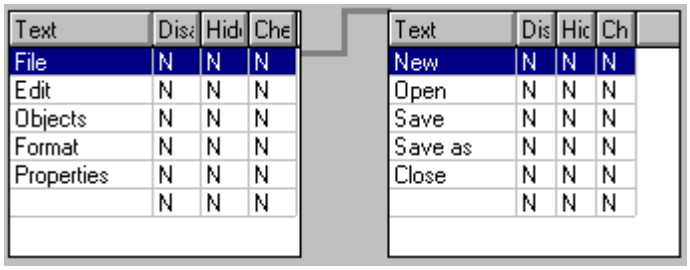
This feature is especially handy, for example, if you are designing multilingual applications.

Creating a Menu

In order to create a menu, it is necessary to place a main menu component, as well as all corresponding sub-menu components. Every entry into a given menu component belongs to the same menu level.

Once you have placed the menu component, the first row in the displayed grid will contain the term 'Menu'. Select the row and begin to key in the new text..

Please terminate every entry with an **ENTER.** This will cause the creation of a new blank row. In case you do not intend to enter an additional menu option, you can just neglect this row; otherwise continue to enter the menu options. All entries into one component will be displayed on the same menu level. If you desire parent-child menu structures, you must place a menu component for each of the siblings.



Connection between a main- and a sub-menu

In this example, two menu components have been placed on the desktop. The leftmost component is intended to represent the parent menu, whereas the right menu component is to become the child menu. This is represented by the connecting line between both components.

You can easily create so called accelerator keys in the menu by prefixing the desired letter with a &-symbol. This will result in the following letter to be displayed with an _ symbol (*underscore*). At runtime, the menu option can be immediately executed from the keyboard by pressing the **ALT** - key along with the appropriate letter. The menu option E̲xit would be executed by pressing **ALT**+**E**.

Please note that the entry of '.' in a sub-menu will result in the displaying of a horizontal line (separator). You can use this facility to group menu options into visible groups.

After you have placed a main menu component and one or more sub-menu components, you can define the hierarchical relationship between these menu components by employing the rubber band technique. Place the cursor above the menu entry which you want to be the main level. Now press the left mouse button and, keeping it pressed, move over the appropriate sub-menu component. By doing so, you will see a rubber band attached to the cursor. Once you release the left mouse button, the two menu components are connected. This is represented with the connection line between the components. Proceed in this manner to define all other menu components.

If you wish to remove a rubber band connection between a menu and a sub-menu, press the left mouse button on the option in the main menu and drag the mouse to a place on the screen other than above a sub-menu. This will remove the previous connection.

The rubber band automatically makes the sub-menu appear when the parent menu option is selected by the user at run time. You do not have to create a link for this procedure.

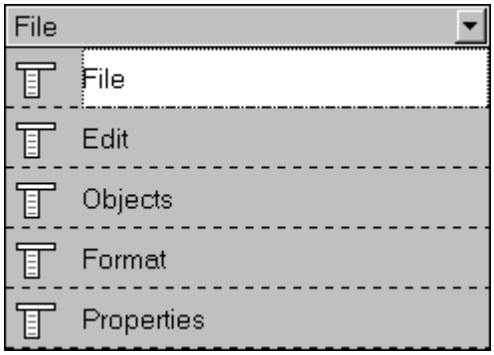
If you wish to edit an existing menu option, select the appropriate entry by clicking the left mouse button on it. Next press **CRTL+E** (edit). Now you can proceed to edit the entry. Once the modifications have been completed, press the **ENTER** key.

In the menu component, you can define whether your menu label should be Disabled, Hidden or Checked. Disabled implies that the menu option is grayed and thus not accessible. If a menu option is set to be Hidden, then it does not even appear in the menu as an option. If a menu option is set to be Checked, then a little check mark will appear, once it has been selected. If it is selected a second time (or the user presses the ***ESCAPE***-Key), then the check mark is removed. This is a commonly used feature in Windows applications and can be viewed in the mindmap **Edit | Set Tab order** option on page 75.

The options Disable, Hidden and Checked are changed at run time via the 'Change attribute'-Message.. (see section Links/Messages/Change attributes on page 270)

Events

The menu component has only one type of event. These are the menu entries themselves. In the above example, these would be as follows:



New events introduced by a menu

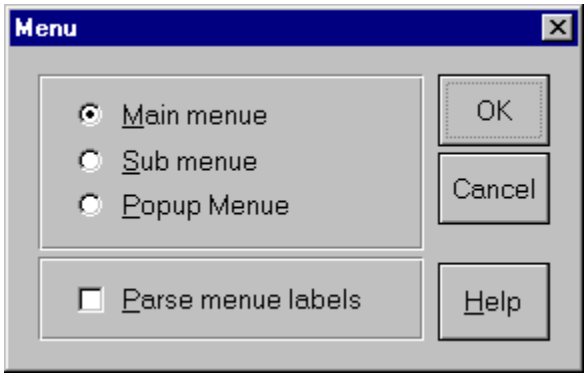
Special Note

When using menus, you should not run a mindmap application in full screen mode. (See also File | Preferences | Application on page 50 about how to run an application in full screen mode)

As the application executes, mindmap menus take effect beginning on the page on which they have been placed. Once a page, which contains different menu components, becomes visible, then these menus become valid, until another page containing menu components is encountered, and so forth. If you choose to place menu components on separate pages, you must assure that the page receives focus. This can be accomplished by jumping first to the page with the menus and then immediately jumping to the page displaying the actual user interface.

Creating a Pop-up Menu

You can specify a menu to be a pop-up menu by selecting the appropriate radio button in the component specific dialog box.

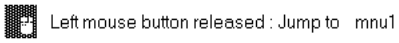


Specific attribute of a menu

There is no need to have the menu on the same page that is currently active while this function is performed.

While normal (window oriented) menus are attached to the caption bar of the mindmap application they are defined in, pop-up menus show up anywhere on the screen, usually placed right besides the current mouse position.

Therefore a pop-up menu is not activated by displaying the page where it has been defined, but rather by issuing a jump link command to the menu object.



This will cause the specified menu to appear as a pop-up menu as close as possible to the current mouse position.

Output Page Component

This component is used to print information. It presents itself as a blank sheet of paper. You can place a mindmap component on the output page in the same manner you would place the component on a mindmap page. The appearance of output pages can be influenced by the user at run time, if intended by the developer.

Since this component is not represented on the toolbox with an icon, you can only access it via the menu **Objects | Output Page**. Once you have selected this component, the appearance of the cursor will change to a crosshair with an attached page. In order to place the component, move the cursor to the screen position where you wish to place the top left corner of the page.

Press the left mouse button and drag the mouse to the size for the page. Then let go of the left mouse button. A white rectangle will become visible. You will be prompted for a name for the page.

The output page component always retains the page metric or size ratio. Thus, you cannot create a page with an invalid height / width relationship.

As the name implies, this component is used to interact with the printer. Therefore, it is important to make the appropriate selection for the paper size (US legal, DIN, etc.). In case you change the size of the frame, the white rectangle inside the frame will still reflect the page metric. If you need to change the position of the component on the screen, grab it by clicking on the frame itself -- not inside on the white rectangle.

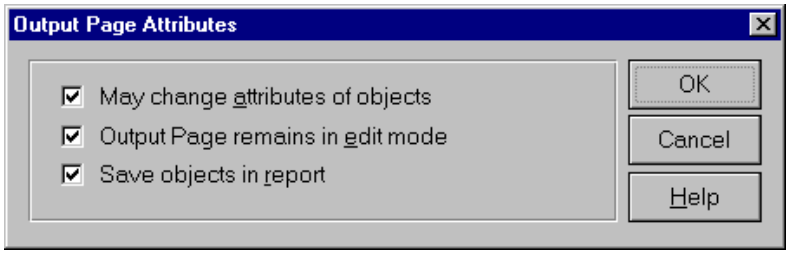
Attributes

Attributes of the output page component

† The common attributes of this component are described in the section Format Menu on page 78.

Component Specific Attribute

If you select this attribute, the following dialog box will appear:



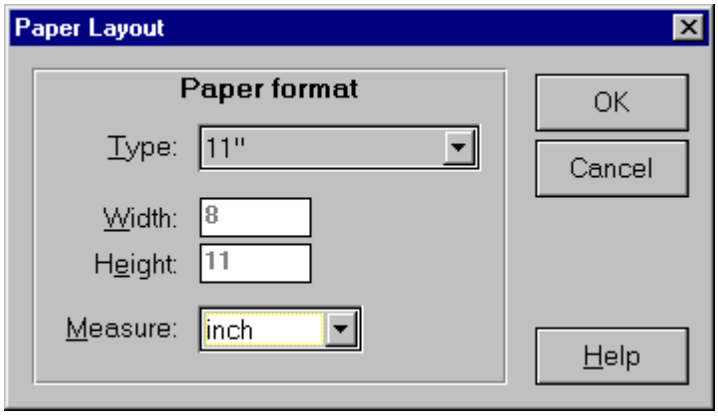
Component specific attributes of an output page

In the component specific attributes of output page components, you can check several options:

- May change attributes of components: This allows the user to change the attributes of components placed on the output page at runtime. This option must be viewed in conjunction with the option Output Page remains in edit mode. If you do not permit editing at run time, the setting of the aforementioned option is invalid.

- Output page remains in edit mode: This option allows the user to change aspects of the output page at run time. By enabling mouse-related drag&drop operations a user may add graphic components or input field components to the output page at run time.
- Save objects in output page: This option should be set when it is desired that components on the output page component are to be stored in the application itself. This should always be set in cases where the user will not be defining the output page at run time.

The size of the output page that you have painted on the screen will not change; however, the actual printed output page will be adjusted to your choice. In this case, you will have to resize the output page component on the screen by hand.

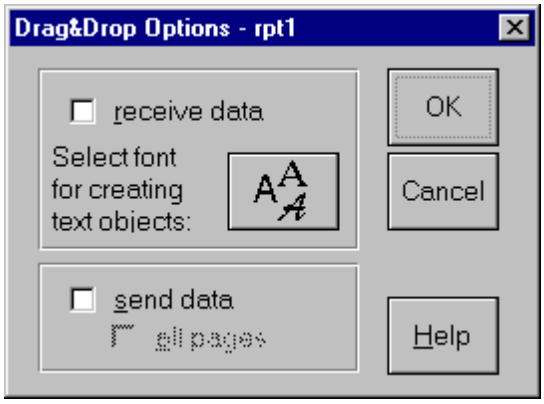


Define the dimensions of the output page

In the Type list, you can choose one of the American / English formats or one of the European DIN formats. Depending on your choice, the Width and Height numbers will be adjusted. By choosing User defined, you are able to key in your own width and height. In the Measure box, you can choose the appropriate unit of measurement.

Drag&Drop

Once you select this attribute, the following dialog box will appear:



Configure drag&drop options specific to the output page component

This dialog box contains an icon for the font attribute. Selecting this attribute will result in the displaying of the common font dialog box.

By means of the drag&drop option, you can have text components created at run time on the output page component.

An input field with the drag&drop option set to send data is required. The output page must have its drag&drop option set to receive data. At run time, the mouse is placed above the input field. Next press the left mouse button and move the mouse to the position over the output page to where the text is to be copied. Once the left mouse button is released, a text component is created and dropped on the output page. It is automatically set to contain a formula and its value is the text contained in the original input field.

Components created in this manner cannot be stored in the application. They only exist as long as the application is in run mode.

➤ See also page 261 for more information about drag&drop.

Additional Events

The output page component does not register additional events.

How to Use an Output Page Component for Printing

You can place a variety of components on an output page, just as you can place them on any mindmap page.

In some cases it does not make sense to place certain components on the output page. For example, placing an input field on an output page does not make sense, since user interaction with it is not possible.

Only the following components can be placed on an output page: text, data table, graphic. The creation of all other components on the output page is prohibited.

If you wish to place text on the page, choose the text component. You can also connect components on the output page to external data sources.

Let's assume you want to place a text component and a bitmap on the output page. Pick up a text component from the toolbox and place it on the output page component. Note that you cannot key in text as you can when placing a Text component on a regular mindmap page.

Text components that are placed on an output page automatically become formula fields. Please open up the formula editor Σ or assign a value to the appropriate text component using a link message (described on page 258).

Σ For this purpose, you have to use the formula function. You can find it by clicking on the Sum symbol in the status bar at the bottom of the screen. Select the text component, click on the Sum symbol and key into the formula field the text you want to see displayed by the Text component. Don't forget to enclose the text in quotation marks, for example 'text'.

➤ See also page 389.

Pick up the graphic import component from the toolbox. Use it to place any bitmap on the output page component. (You'll find several bitmaps in the Windows directory of your system, for example).

Now let's place two other components on the mindmap page, right beneath the output page component. First, place one command button on the mindmap page. Next, click on the in-

put/output button of the toolbox and click on the printer symbol of the input/output dialog box. Don't worry about the other options in this message box; just click the OK button to accept the settings and close the box. Now, place the printer component on the mindmap page (not on the output page component) and key in a name when you are asked for it by mindmap.

- The printer component is described on page 207.
- The command button is described on page 125.

Now, let's add the printing functionality. Select the command button, activate the attribute toolbox and click on the links button. Create a new link with the event Left mouse button released and the message Drag&Drop. In the Move Drag&Drop Data from section, click on the symbol of your output page component. In the Move Drag&Drop Data to section, click on the symbol of your printer component. Close the dialog box by clicking on the OK button in the upper right corner. The resulting link should be:

Left mouse button released: `Drag&Drop rep1 → prt1`

Now, change to the run mode of mindmap and click on the command button. If everything has been done as described, your printer will print out a page with the text of your text component and the bitmap.

Depending on how the application has been designed, the user might actually never see the output page. Usually, the print functionality is executed by simply pressing a button. The application never visibly jumps to the page containing the output page.

Special Remark

Text components and data table components that have been placed on an output page, will print on subsequent sheets of paper if there is not enough space to print their contents on the first page. Data tables automatically wrap to the next page. Text components must have their Auto print multiple pages attribute set.

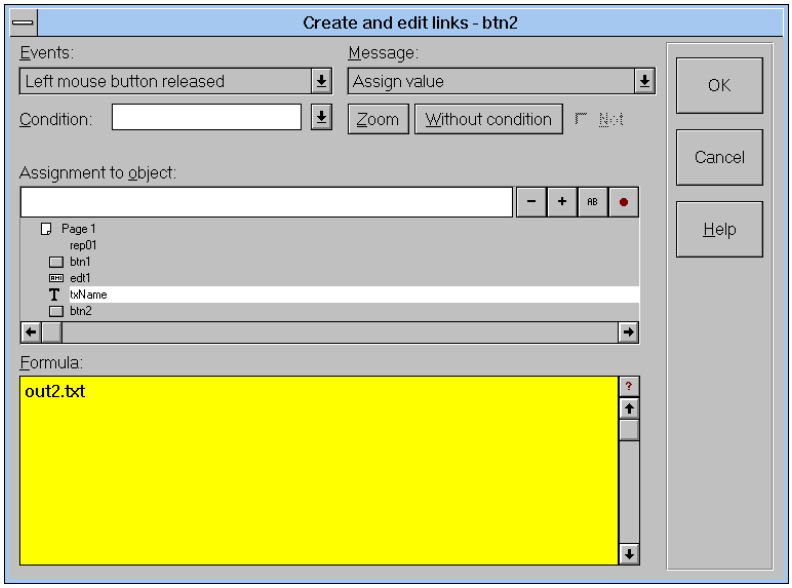
- See **Text Component** on page 134.
- See more information about the parser on page 388.

It is possible to use components in formulas, which have been placed on output pages. The notations

- <report name>.<component name> or

- <report name>:<component name>

are references to a particular named component on an output page. Following is an example of how the content of a text component on an output page is referenced by a text component on a mindmap page:



Properly address components on output pages in formulas

MCI Component

Based on the MCI (Media Control Interface) specifications, mindmap offers a multimedia component that serves as a vehicle to display and maintain Video for Windows (AVI) files, Sound (WAV) files, as well as other types of multimedia devices in mindmap applications.

To create a multimedia component, select the appropriate icon from the toolbox or in the menu **Objects | MCI Object** and draw the component at its desired position and size. Since all multimedia devices share a common interface, this interface is implemented as mindmap link messages, as far as commands to the multimedia object are concerned. A- component specific at-

tributes dialog to alter the appearance of the component is available, as well. This feature allows you to open all types of MCI components. In the select dialog box, all the file extensions are listed that can be loaded. They are as follows: '*.wav; *.mid; *.rmi; *.avi; *.mmm; *.mov; *.pic; *.jpg; *.flc; *.fli; *.aas and *.mp2'. In the dialog box, only the files with these extensions will be listed.

The mindmap MCI component relies on the existence and functionality of special MCI drivers. Make sure that these drivers are properly installed before you create an MCI component.

If you are running mindmap on Windows 3.1, 3.11 or Windows for Workgroups you must install Video for Windows (VfW). Refer to the corresponding documentation supplied with Video for Windows to learn how to install it. If you are running mindmap on Windows 95, VfW is part of the operating system. Make sure that it has been installed through the Add/Remove Software applet in the control panel.

If you choose to install additional multimedia software which interfaces with MCI, you will possibly be able to interact with more file types.

If you have placed an MCI component, e.g. an AVI file, it will display itself as follows:

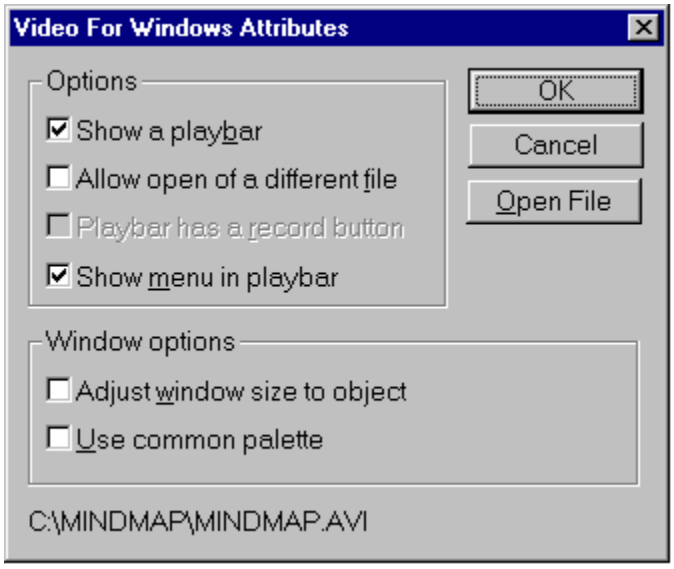
A video as an example for an MCI component

Attributes

Attributes of the MCI component

† For additional information concerning the common attributes see the section Format Menu on page 78.

Component Specific Attribute



Specific attributes of the MCI component

The following options are defined through the component specific dialog:

Option	Description
Show a play bar	Defines that the object should have a play bar and at least a start / stop button.
Allow open of a different file	Defines that the user is allowed to open a new multimedia object through an open command in the menu (if it exists).
Show menu in play bar	Defines that a pop-up menu should appear if the right mouse button is

	pressed over the surface of the object or, if a play bar exists, a menu button should be visible.
Play bar has a record button	Defines that a record button should appear for devices that are capable of recording information (e.g. wave audio).
Adjust window size to object	Defines that the size of the multimedia object is controlled by the size of the mindmap component instead of being auto-sized depending on the size of the video source.
Use common palette	If this option is checked, the common palette defined by mindmap is selected for the Video for Windows object, instead of letting VfW define its own palette. Setting the graphic object to use a common palette as well, is useful if a video has to be played concurrently with a graphic on the same screen. Note that this applies only to 256 color palletized graphic modes. Also note that specifying a common palette may result in some loss of visual quality.

Additional Events

The links dialog box for multimedia components has some of the standard events (see page 244). The additional events are shown in the following table:

Event	Description
Mode changed	This event is generated if the mode of the multimedia component changes (e.g. playing, seeking, stopped, ...).
Device not ready	This event is generated if the device is in a state where playing a multimedia file is not possible, as is the case if a CD was ejected from a CD drive.

Stopped	This event is generated when the device ends playing a multimedia sequence, regardless of whether the device stops through user interaction or when the end of the media is reached. Prior to sending this event, the Mode changed event has been sent.
Playing	This event is generated when the device starts playing a multimedia sequence. The event is sent after the Mode changed event has been fired.
Recording	This event is generated when the device starts recording a multimedia sequence. The event is sent after the Mode changed event has been fired.
Seeking	This event is generated when the device moves to a new position of a multimedia sequence. The event is sent after the Mode changed event has been fired.
Paused	This event is generated when the device is in pause mode. The event is sent after the Mode changed event has been fired.
Device is open	This event is generated when a multimedia sequence is ready to be launched after loading. Prior to sending this event, the Mode changed event has been sent.
Position changed	This event is generated continuously once every second to indicate the progress of playing a multimedia file. Use the mciGetPosition or mciGetPositionString functions to retrieve the current position. These functions are described on page 472.
Size changed	This event is generated when the MCI interface changes the size of the display window.
Media changed	This event is generated when a new media (such as a CD) is loaded.
Error	This event is generated when the MCI interface detects an error. The formula value of the MCI component contains the appropriate error string.

Visual Basic Component

This component allows the integration of Visual Basic custom controls. These so called VBX controls are embedded in mindmap, similar to other mindmap components, with a common user interface and messaging protocol. Depending on the capabilities of a particular VBX control, only the command set (here called properties) varies from control to control. Since mindmap does not distinguish between the Visual Basic type of run and edit mode, from the control's point of view, the system appears to be in run mode all the time. Please note that controls that behave differently when in VBX edit mode, may not work properly in mindmap.

In so far as VBXs have been installed, (See **File | Preferences | VBX** on page 56), you will see their graphical representation on the toolbox, given that they have registered themselves with an icon. You will also see them entered on the menu **Preferences | VBX**. Once you have selected a VBX component, the cursor changes to a crosshair with the letters VBX attached. Move the mouse to the screen position where you wish the VBX to appear. Press the left mouse button and drag the mouse until the dashed rectangle has the desired size. Release the mouse button.

Installing a VBX Custom Control

VBX files are attached to mindmap by selecting them in the preferences dialog box **File | Preferences | VBX**. Once installed, you must shut down mindmap and restart it, in order to load the new control into the menu and toolbox and to make it accessible (See also page 56). Every installed VBX has at least an entry in the menu. Generally, VBXs also register an icon on the toolbox.

mindmap emulates portions of Visual Basic. To be installed into the mindmap environment, it is required for a VBX to comply to the Visual Basic API Version 1.0. In general, if a VBX uses functionality of Visual Basic Versions 2.0 or higher, this par-

ticular VBX cannot be installed in mindmap. In this case it either displays a dialog box or it may behave unpredictably.

Since the environment appears to be in run mode from the VBXs perspective, the VBX must support dynamic creation in run mode. Some VBXs require they be inherited from a Visual Basic Form when Visual Basic switches to run mode. These VBXs might not behave properly in mindmap.

Also, since mindmap's drag&drop metaphor is completely different from Visual Basic's, drag&drop functionality supplied by a VBX is not supported in mindmap.

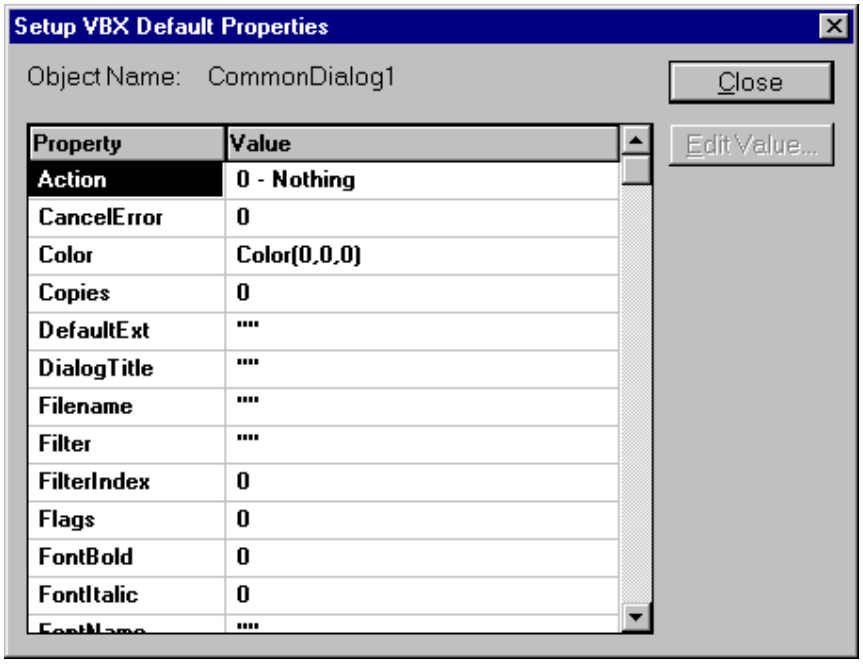
Attributes

Attributes of a VBX component

† For further information regarding common attributes see the section Format Menu on page 78.

Component Specific Attribute

If you select this attribute, the following dialog box will appear. The actual entries in the list vary depending on the VBX:



Example for component specific attributes of a VBX control

All properties that have the PF_RunmodeW and PF_RunmodeR bits set, can be edited through the VBX's component specific attributes dialog. This dialog is common to all VBX controls. The left column lists the Property name in alphabetical order and the right column contains the actual Values. These values may be set by double-clicking the left column (i.e. the property name) or by simply typing the desired value. Some properties allow you to make the value selection from a predefined list. You can recognize this if the Edit Value... button is activated when you select a property. Please note that all property values are parsed. This means that parser statements containing the names of other components or mindmap parser functions may be used. Be sure to surround strings with double quotes.

Once a property is edited, the VBX control should reflect the change, even if this dialog is not closed.

Also note that all properties set through this dialog are saved into the application file and automatically restored at load time.

Of course, you may alter the properties of a VBX component at run time by creating a link and specifying what property should receive which value. Similar to most other mindmap components, VBX components also register a set of events which appear in the list of events in the link dialog, at the bottom of the list. Please note that all events that a VBX might execute are listed, but that some events may appear in this set which do not work in the mindmap environment (e.g. dragging events).

The VBX controls can't be described in more detail, because there are thousands of controls available. For more details on the controls, you should consult the manual of the particular VBX that you wish to use. Also note that mindmap can only change or modify those aspects of the VBX's properties that the original VBX developer has allowed us to access.

Properties

Communication with VBX Controls is accomplished by getting and setting properties. Property types are either:

Strings

Strings are null-terminated ASCII strings compatible with the concept of mindmap. They are limited in size to 16383 bytes.

Numeric

All numeric types supported by VBX controls are converted to and from the only number type used in mindmap.

Enumerations

Enumerations are numbers as well. For the user's convenience, their value may be selected from a list in some of the dialogs.

Pictures

Pictures are currently not supported.

Additional Events

Similar to most other mindmap components, VBX components also register a set of events which appear in the list of events in

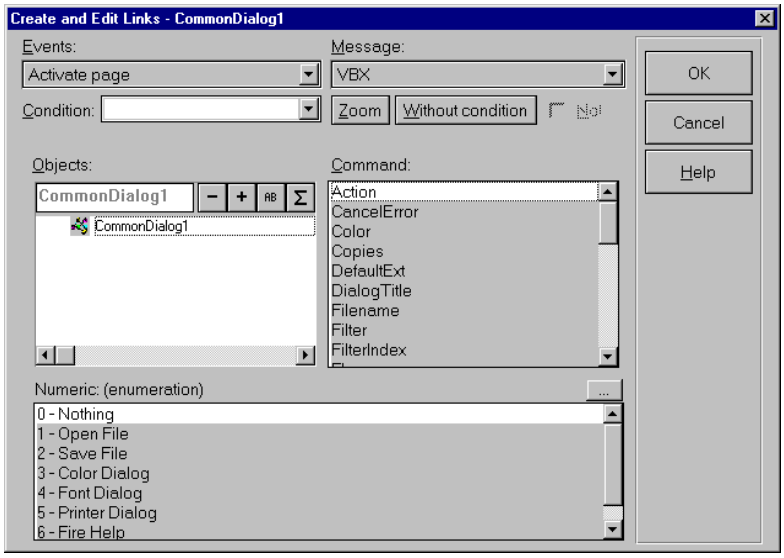
the link dialog. Please note that all events that a VBX might execute are enumerated, so that some events may appear in this set which do not make sense in the mindmap environment (e.g. dragging events).

Some VBX implementations may pass arguments via events. Please use the GetParam function registered by the VBX component to access these parameters.

➤ Refer to the parser documentation on page 401.

Communication

Of course, you may alter the properties of a VBX component at run time by creating a link and specifying what property should receive which value.



This link message controls the various properties of a VBX control

The figure displays a link on the command button. If the button is clicked, attributes will be changed.

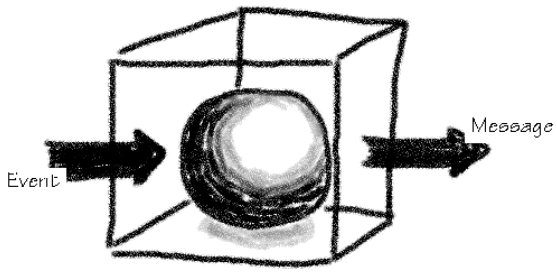
The link dialog box for the messages relating to the VBX contains properties, which are displayed as commands. You enter the values in the lower portion of the dialog box.

Chapter 6 Links

General Overview

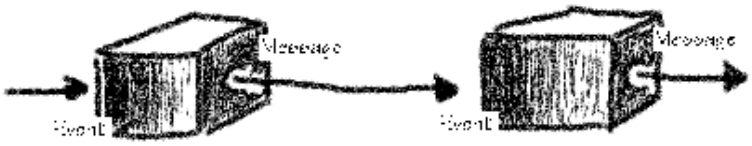
Some Background Considerations

Defining a behavior for components in mindmap is equivalent to the process of programming an application in a conventional development environment such as C++, VisualBasic, PowerBuilder, etc. In mindmap, defining the behavior is accomplished by assigning one or more links to a component. A link is comprised of exactly one incoming event and one outgoing message. This pair is what we call a link. The process of defining such links in mindmap is referred to as linking components.



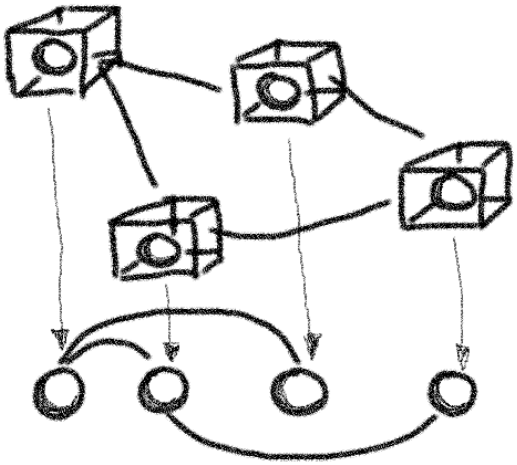
mindmap's event - message model

An event functions as a trigger for a component. When a trigger occurs, the component which has been associated with the trigger will receive control and do as it has been instructed in the message. It will generate the message. A message, in turn, will cause a specific reaction by the associated component or it will act as an event for a subsequent component.



One component passing a message to another component

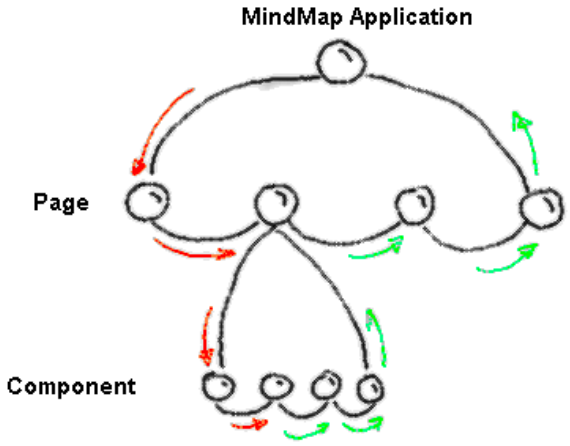
Thus, building an application can be viewed as constructing a network of components, sending and receiving both events and messages. In the above model, a component is receiving exactly one event and generating one message. In other words, it has exactly one link associated with it. If this were always the case, an application would be fairly limited. Therefore, mindmap allows multiple links to be assigned to a component.



A network of *mindmap* components

In a two-dimensional representation, the cubes represent the components and are displayed as a network, albeit in this case a very simplistic one. If the two-dimensional representation is collapsed into one dimension (represented by the balls), then an application can be viewed as a single list of components connected in a parent-child relationship.

This leads to a very important aspect of mindmap, which should be kept clearly in mind when constructing applications — the mechanism by which messages are passed in the system. Let us take this, and granted, a somewhat abstract representation, one level further. Consider the balls in the above representation to be pearls on a chain. In this little example, the application would consist of one string of pearls. Let's look at a somewhat more complex application.



Message chain

This is a necklace of necklaces — where multiple strings of pearls are interconnected. The top level of this picture represents the application itself. At this level, events are received from the operating system. When a user clicks on the mouse, the operating system intercepts the event and determines which programs are running at the time. It then determines which program is in foreground and should presumably be waiting for user interaction. Once this has been determined, the operating system forwards the event into the message handling facility of the program, in this case mindmap.

Next, mindmap – actually its event-message handler – grabs the event and forwards it to the first page. All pages are on the same string of pearls, meaning that they are logically at the same level. The event is passed into the necklace and the first page receives it. The page determines whether or not it is to act

on it. In case it decides it doesn't want to deal with it, it passes the event on to the next page on the necklace. The process begins anew. Eventually either a page decides to process the event or it ends up back at the top of the chain, in which it is simply discarded.

If a page decides that it should deal with the event, it grabs the event, opens up its little necklace of components on the page and forwards the event into this chain. The process begins again. The first component on the page takes a look at the event and decides if it wants to act on it. If not, it passes the event on to the next in line. Either the event is squeezed back up to the page level without having been dealt with, or some component on the page has grabbed it and acted on it. In this case, the event will be flagged so that subsequent components will not deal with it. The flagged event is then passed through the component level of the chain, up to the page level, where it is then rushed back up to the application level.

The sequence for processing events is – graphically speaking – either left to right or right to left, depending on the particular type of event. Well, how does this relate to the components as viewed on the screen? Easy. The first page in a mindmap application is the leftmost page. The next created page will be 'next in line', or to the right of the previous page. Inserting a new page can be envisioned as cutting open the necklace, adding in a new pearl and knotting the chain back together again.

The same metaphor can be used for the components themselves. The first placed component is first-in-line. It is at the leftmost position on its segment of the necklace. The next created component is attached to the right of its predecessor.

This 'pearls-on-the-necklace' order can also be viewed differently; namely, in the sense of foreground and background. The first component placed on a page is in the background. The next component is in foreground, relative to the first, and so on. Thus, the last placed component is in foreground, relative to all other components on the page. If you now move a component into background (via the menu option **Edit | Background** or vice versa **Edit | Foreground**, see page 67), what you are actually doing is rearranging the pearls on the necklace. You are, figuratively speaking, cutting open the necklace and inserting the component into a new position.

Applying this metaphor to groups, we get the following picture: When you create a group of components, what is really happening is that the component necklace segment is cut open, a new type of component is created and inserted (being the non-visible group component) and all components belonging to the group are strung onto the new segment, which dangles off the component segment. A group of groups is, thus, a necklace of necklaces.

Keep in mind, that the order in which the components receive and process their messages, also corresponds to the order in which they are painted onto the screen -- from background to foreground or from foreground to background.

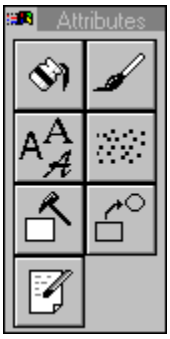
The direction of event passing actually depends on the type of event. Generally, all mouse-related events are passed from right to left (foreground to background) in order to make sure that events being in the foreground receive mouse clicks first, as one would expect. On the other hand, system related events like `Application started` or `F3 key pressed` are passed from left to right.

We strongly recommend that links acting on the same type of system related event (like `Application started` or `Page activated`) are kept together on a single component. This makes it easiest to keep control over the sequence of operation.

How to Define a Link

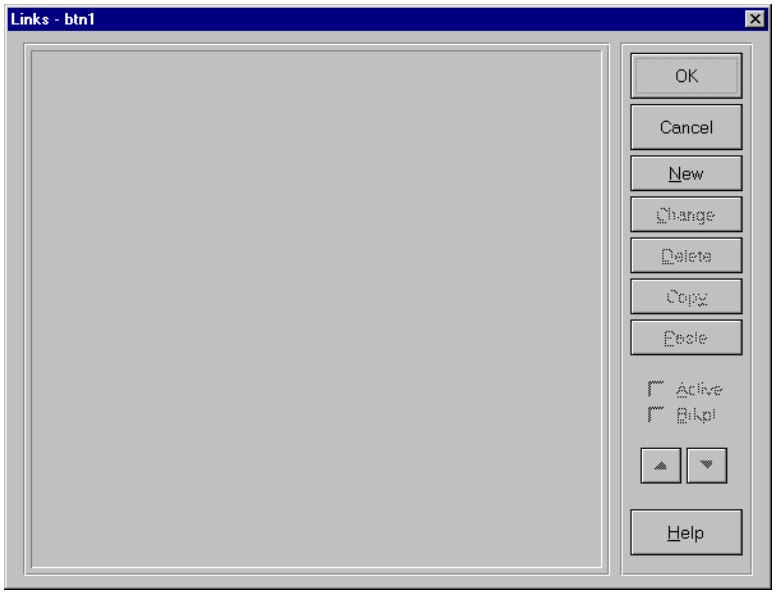
Before getting into the details of what each event and message actually does, let's look at how they are assigned to a component.

Select a component to which you want to assign a link. Next, either click on the menu option **Properties | Links**, press the **F6** function key, or activate the attribute toolbox for the component. Let's assume you did the latter.



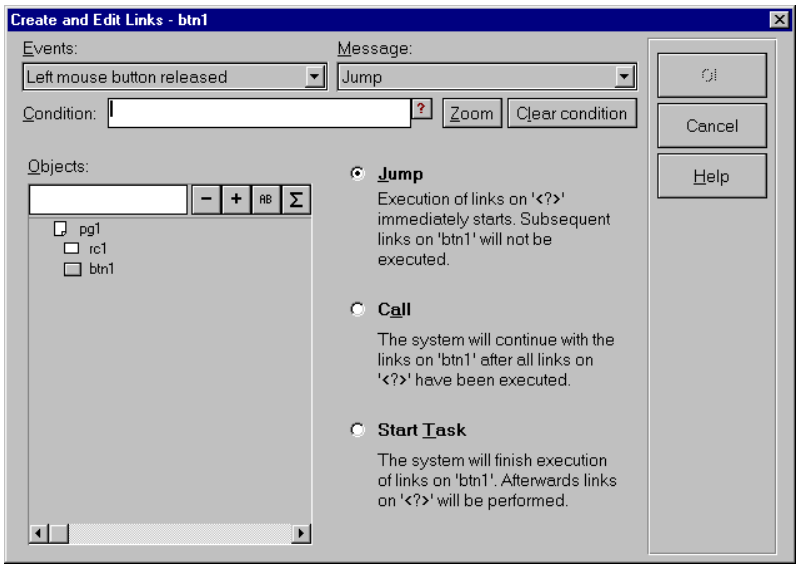
Attributes of a component

Now, click on the icon displaying an arrow leaving a little square and pointing towards a little circle. This will result in the following screen being displayed:



This dialog will list all links that are defined for a selected component

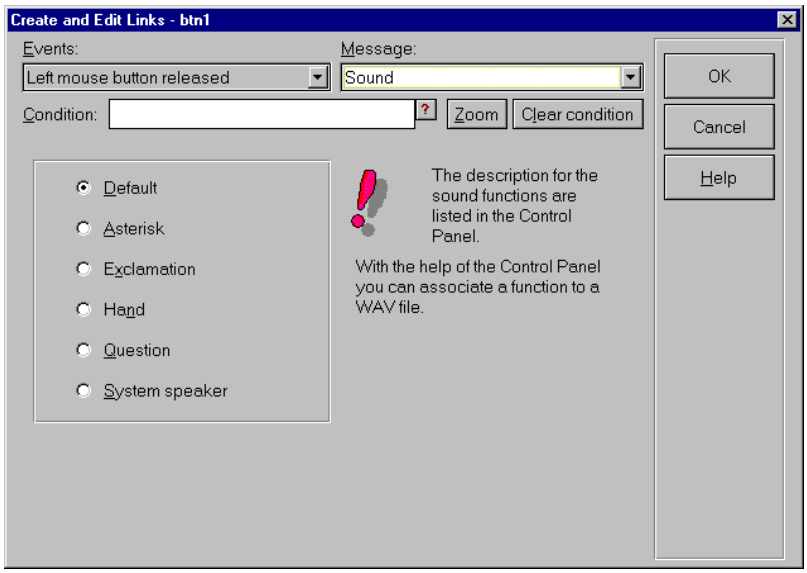
Since no links have been defined so far, the major portion of this dialog box is empty. In order to define a new link, click on the New button on the right side of the dialog box. This will open up the following dialog box:



Multipurpose dialog box to create and modify links

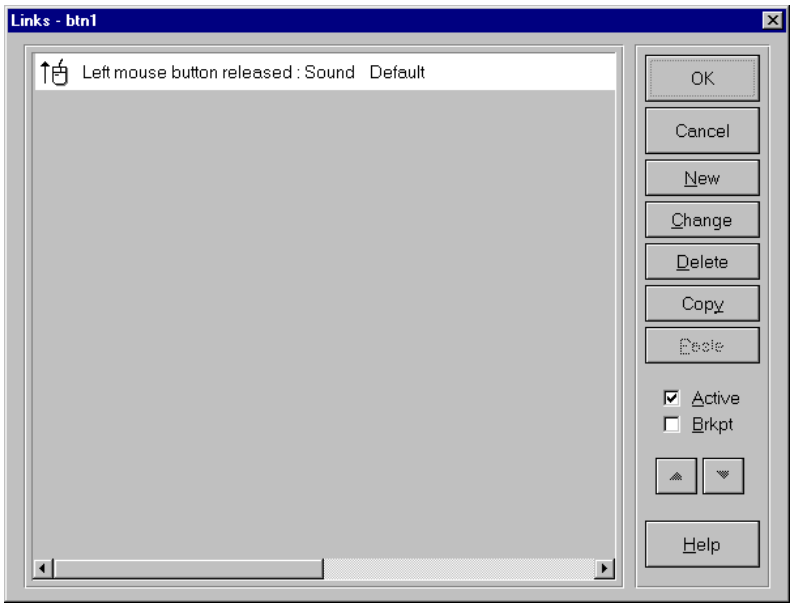
This is the control center. Here is where you actually define the behavior of a component. At the top left of this screen, you see the Events list. Next to it is the Message list. Underneath both lists is an area which deals with conditions. The rest of the screen is dedicated to whatever definitions are necessary, in conjunction with the selected event-message combination. Therefore, it will change its appearance depending on the selection made earlier.

To demonstrate the method by which a link is defined, drop down the Events list. Select one of the events, such as Left mouse button released. Next, drop down the Message list and select the message entitled Sound. This will result in the following screen:



A different type of link - play a sound

As you will notice, the bottom section of the dialog box has changed to reflect the selection of the Sound message. Click on any one of the options on the left side and accept the selection by clicking on the OK button on the right side of the screen. This will result in returning you to the initial link dialog box, only this one has now been updated to reflect the new link definition.

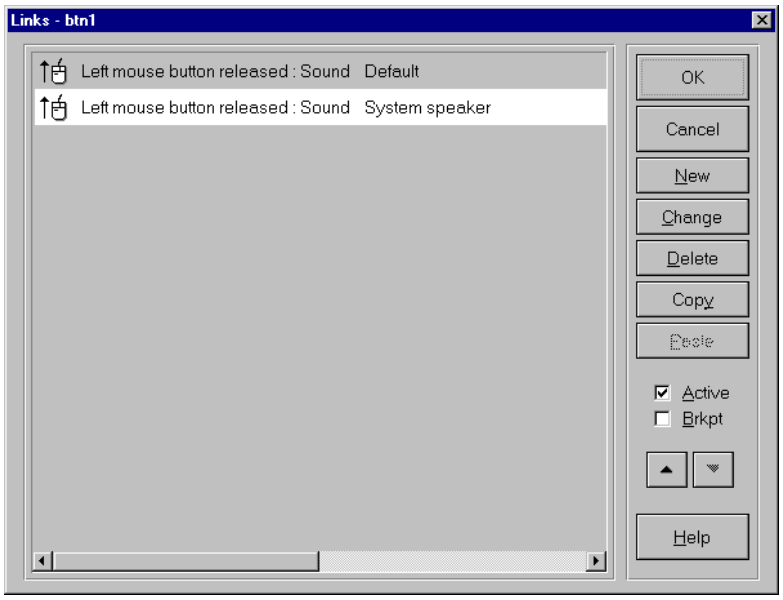


A link has been created for the component btn1

The dialog box now displays a link defined for the component **btn1** (see the caption bar on the dialog box). The link can be read to say that:

'When the mouse has been moved on top of the component btn1, and the left mouse button has been clicked and subsequently released, the sound associated with the default setting will be produced.'

Next, create another link, this time associating a different sound with the same event. The screen should then look similar to this one:



Two links are defined for the button btn1

If you were to execute this little application, then the following actions would take place. Once you clicked on the push button named btn1, the sound associated with Default would be played (provided you have a sound card installed), immediately followed by the sound associated with System speaker. In other words, the processing sequence of links is top-to-bottom of the list for any given component. There is one aspect you should be aware of, though. Some events are generated sooner than other events, an example being `Application started`. If you have assigned this event to any component in the application, then it will be executed, and thus the message associated with it will be performed, before the list of any component will be processed.

Relative to a component, the link list is processed from top to bottom. Relative to the application though (which in some cases might override the apparent sequence associated with any given component), certain events are generated without user interaction, and prior to any user interaction events. These will be processed first.

If you wish to rearrange the order of link processing, you can easily do so. In this simple case, let's assume you want to hear

the sound associated with the System speaker entry before the Default sound is generated. In this case, select the second link (Sound Speaker sound). It will invert the background color. On the bottom right hand side of the dialog box, you can see two arrow heads, one pointing up, the other pointing down. Click on the one pointing up. This will cause the second link to be moved up, in front of the Sound Default link.

The mouse cursor indicates an insertion point

Another way of rearranging the links is by selecting the link you wish to move. Then, keeping the Left mouse button pressed, drag it to the position you wish it to be moved to. The cursor will change its appearance to display where it will accept the placement of the link. This cursor will appear whenever the mouse has been positioned between two existing links, or at the top or the bottom of the list of links.

You can also delete, copy, and paste links within this screen, as well as between links screens.

If you wish to delete a link, select it and then click on the Delete button on the right side of the dialog box. If you want to delete multiple links, you can select them either individually by keeping the **CTRL** key pressed and clicking on the link, or by keeping the **SHIFT** key pressed and clicking on the links. Using the **CTRL** key approach permits you to select non-contiguous links. Using the **SHIFT** key method will select all links between the first and the last selection made using the mouse.

Using the Copy button on the right side of the dialog box, you can place the selected link(s) on the clipboard. You can employ the same selection methods just described for the delete function. Once selected, the links are placed on the clipboard as soon as you click on the Copy button. Copying links to the clipboard will overwrite any links placed there previously.

You can retrieve the links from the clipboard by clicking on the Paste button. The link(s) off the clipboard will be placed immediately above the links selected at the time. Thus if you already have to links on the screen and you insert the contents of the clipboard, then assuming the first of the two links is selected, the contents will be placed as the first links.

Another feature available is that of toggling a link active and inactive. Links are active by default. If you wish to temporarily turn one off (make it inactive), then simply select it and mark

the check box on the lower right hand side of the dialog box. The selected link(s) will be grayed and will not be executed during run time.

Events

Common Events

All components have a common heritage in regards to the events they support. When a component developer builds a new component he/she is offered a set of generic events. It is then the decision of the developer to support all, or any sub-set, of these events. In addition, the developer can then register new events, that are specific to the component being developed.

The simpler components do not support their own events. They merely inherit their events from the generic list of events. Following is the set of generic events. In general, all – or at least a major portion of them – are supported by all components:

Many users have difficulties performing mouse double click operations. This is in part due to the sensitivity settings for the mouse. When attempting to *double click* many users move the mouse a few pixels. This is often enough for Windows to interpret the action as two separate single mouse clicks. It is generally advised to avoid using this event, if at all possible.

Icon	Event	Description (When generated)
	Left mouse button clicked	When left mouse button is pressed and before it is released.
	Left mouse button released	When left mouse button has been released. This (and Right mouse button released) is the preferred event on which to trigger buttons.
	Left mouse button double clicked	When left mouse is clicked twice in rapid sequence.
	Right mouse button clicked	When right mouse button is pressed and before it is released.
	Right mouse button released	When right mouse button has been released.

Icon	Event	Description (When generated)
		This (and Left mouse button released) is the preferred event on which to trigger buttons.
	Right mouse button double clicked	When right mouse button is clicked twice in rapid sequence.
	Activate Page	When control is passed to current page (jumped to or to component on page). Use this event when you wish actions to be taken before anything else on the page, to which is being jumped, is to be executed.
	Deactivate Page	When control is passed to another page. Use this event if you need to 'clean up' something, before turning control over to the next page.
	Mouse movement	Any movement of the mouse will generate this event. This event is sent to the component which the mouse is currently above. Use this event very cautiously, since it will be generated anytime the mouse is moved within the border of the selected component.
	Mouse movement into object	When the mouse cursor is moved into selected component. Use this event in conjunction with the changing of the appearance of the mouse cursor.
	Mouse movement out of object	When the mouse cursor leaves the selected component. Use this event in conjunction with the changing of the appearance of the mouse cursor.
	Application started	When the application is started (put into run mode or executed as EXE file).

Icon	Event	Description (When generated)
	Application terminated	When the application is normally terminated (put into edit mode or quit as EXE file), this event will be generated. This event is often used to clean house (i.e., set variables to some predefined value) before leaving the application.
	Goal of a jump	When the selected component has been jumped to (used in conjunction with sub routines/ sub assemblies).
	Error in calculation	When an error in evaluating a formula is encountered. This event is sent to all components on the active page. Parser error messages are described on page 390.
	Begin Drag&Drop	When a drag&drop operation using the mouse begins. This event is sent to the component initiating the drag&drop operation.
	End Drag&Drop	When a drag&drop operation using the mouse ends. This event is sent to the component that has initiated the drag&drop operation.
	F1-**F12**-Keys	When any function key is pressed in run mode for the selected component. This event is sent to all components on the currently active page. **F4** toggles back into edit mode, if the file is running as an .MM file. When running an .MM file which has been converted to an .EXE file, this function key will not toggle between edit mode and run mode.
	**Esc**-Key	When the **ESC** key is pressed for the selected component.

Component Specific Events

The following list contains those events which are available if, and only if, the associated component has been installed. These events are specific to the particular component. They will only be visible in the list of events for a component of the same type.

Buttons

Icon	Event	Description (When generated)
	Button pressed	When a command button has been pressed and before it is released.
	Button released	When a command button has been released.
	Row up/left	When Up arrow has been clicked on a selected scroll bar component.
	Page up/left	When area on selected scroll bar between elevator and the Up arrow has been pressed.
	Row down/right	When Down arrow has been clicked on a selected scroll bar component.
	Page down/right	When area on selected scroll bar between elevator and Down arrow has been clicked.
	Move	Only when the elevator bar on the scroll bar is moved up and down (or left and right on a horizontal scroll bar), this event will be generated.
	Scroll change	The event is generated when either the elevator bar is moved, one of the arrows has been clicked on, or the area between the elevator bar and an arrow has been clicked.

➤ Read more about buttons on page 125.

Data Tables

Icon	Event	Description (When generated)
	Receive focus	Whenever the cursor is placed inside a cell of the data table, this event will be generated. This event is only generated when the data table itself receives focus. Use the event Cursor changed if you want to determine each single click into a cell of the data table.
	Lose focus	Whenever the data table itself or a cell inside the data table loses focus, this event will be generated.
	Cursor changed	Anytime the cursor is moved to another cell within the data table, this event is generated.
	Double-click	Whenever a cell in the data table receives a double click, this event is generated.
	Abort edit mode	This event is sent when the cell edit mode is canceled with the **ESC** key.
	Begin edit mode	The normal procedure to begin the edit mode in the cell of a data table is to press the **CTRL+E** key. This will generate the event. If the cursor is placed inside a cell of the data table and the user begins to enter in characters, the event will also be generated.
	Cell changed	Whenever the content of a cell has been changed, this event will be generated. Keying in the same characters as are already in the cell, will not generate this event.

	Order changed	If the attribute, which permits the user to rearrange the rows at run time, has been set, then the event will be generated whenever a row has actually been rearranged. The process for rearranging rows is to select a row, keeping the Left mouse button pressed, moving it to another row and letting go of the left mouse button. The row can only be 'dropped' when the cursor changes its appearance, signaling a valid position.

➤ More information about data tables can be found on page 184.

Input Fields

Icon	Event	Description (When generated)
	Enter	If the cursor is inside the input field (input field has focus) and the user presses the **ENTER** key, this event is generated. If the input field is set to display multiple lines, then **ENTER** will cause carriage/line feed and the cursor will be placed into the first column of the next row.
	Keyboard input	If the cursor is inside the input field (input field has focus) and the user begins to type in characters this event is generated.
	Receive focus	Whenever the cursor is placed inside the input field, this event is generated.
	Lose focus	If the input field had focus and the cursor was used to click somewhere other than inside the input field, this event is generated.

	Cursor changed	If the cursor is inside the input field (input field has focus) and the user moves the cursor (caret) to another position by clicking on the arrow keys on the keyboard, this event will be generated. Moving the position of the caret by means of the mouse will not generate this event.

➤ The input field is described on page 193.

MCI

Icon	Event	Description (When generated)
	Mode changed	This event is generated, if the mode of the multimedia component changes (e.g. playing, seeking, stopped, ...).
	Device not ready	This event is generated, if the device is in a state in which playing a multimedia file is not possible, as is the case when a CD was ejected from a CD drive.
	Stopped	This event is generated, when the device ends playing a multimedia sequence, regardless of whether the device stops through user interaction or if the end of the media is reached. Prior to sending this event, the Mode changed event will be sent.
	Playing	This event is generated, when the device starts playing a multimedia sequence. The event is sent after the Mode changed event has been generated.
	Recording	This event is generated, when the device starts recording a multimedia sequence. The event is sent after the Mode changed event has been

		generated.
	Seeking	This event is generated, when the device moves to a new position of a multimedia sequence. The event is sent after the Mode changed event has been generated.
	Paused	This event is generated, when the device is in pause mode. The event is sent after the Mode changed event generated.
	Device is open	This event is generated, when a multimedia sequence is ready to be launched after loading. Prior to sending this event, the Mode changed event will be sent.
	Position changed	This event is generated continuously once every second to indicate the progress of playing a multimedia file. Use the mciGetPosition or mciGetPositionString functions to retrieve the current position. See also page 472.
	Size changed	This event is generated, when the MCI interface changes the size of the display window.
	Media changed	This event is generated, when a new media (such as a CD) is loaded.
	Error	This event is generated, when the MCI interface detects an error. The formula value of the MCI component contains the appropriate error string.

➤ More information about the MCI component can be found on page 221.

VBX

If you register a VBX, then it will usually augment the list of events by itself. Since these events are specific to the component, there is no way to document, here, which events will be registered. As is the case with all other component specific events, the events registered by the VBX will be appended to

the list. Some of the otherwise available events might be excluded from the list, since the VBX paradigm does not support them (such as drag&drop) between non-VBX components.

You should refer to the documentation accompanying the VBX in order to determine which events are/will be available.

Encapsulation (*Client / Server*)

An encapsulation component will always register at least one event. In addition to this event, the developer of the encapsulation component (subassembly) will be responsible for exposing other facilities. These facilities, otherwise referred to as the sub-assembly's API (Application Programming Interface), will appear in the event list, as defined by the component developer:

Icon	Event	Description (When generated)
	Server has closed	This event is generated when the server has terminated normally. The client connected to this server will receive this event and can process it.

Database

The following events are generated by any installed database:

Icon	Event	Description (When generated)
	Record loaded	When a query is sent to a database, and a record has been returned to mindmap, an event of this type is generated. Also, if the database cursor has been moved to a new position, by means of the First record, Previous Record, Next record, Last record or Go to record commands.

	Delete failed	When the mindmap application attempts to delete a record in the database and the operation has failed, this event is generated. A possible cause for this failure might be that the database is write-protected.
	Insert failed	When the mindmap application attempts to insert a record in the database and the operation has failed, this event is generated. A possible cause for this failure might be that the database is write-protected.
	Update failed	When the mindmap application attempts to update a record in the database and the operation has failed, this event is generated. A possible cause for this failure might be that the database is write-protected.

➤ More information about the database component can be found on page 146.

Input/Output

The following events are generated, independent of the type of data stored on the clipboard.

Icon	Event	Description (When generated)
	New data in clipboard	Whenever something is copied to the clipboard, this event is generated. This event can be used to signal to the application that a copy has been successful.
	Clipboard empty	Whenever another application has changed or has cleared the contents of the clipboard.

➤ Read more about the input/output component and the clipboard on page 206.

Menu

Icon	Event	Description (When generated)
	'Menu'	When a menu component is placed on the screen and a text has been assigned to a field, then this text will appear as the event option in the event list. The text string itself has no meaning. It merely functions as a placeholder representing the n^{th} menu entry. Note: The only events a menu component has, are those entered into the component itself, by the developer.

Messages

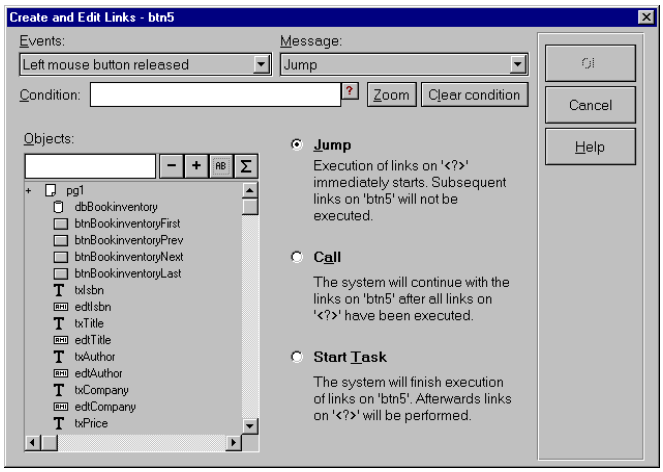
Common Messages

The following list contains all the messages which are available in the standard environment. This means that even if you have not installed additional components, these messages will be available to you. Each additional component can choose to register its own messages. In this case, they are added to this list. The description of the component specific messages are dealt with on a component-by-component basis at the end of this list.

Whenever you select a message from the list on the right side of the dialog box, the lower section of the dialog box will change to reflect the selection. The actual contents of the link dialog box after having selected a message is, thus, a function of the selected message.

Jump

Once you select the jump message, the dialog box will take on the following appearance.



This dialog lets you jump to a different component

This message is used in a number of different ways. Its most common use is to transfer control from one page to another. By no means is this its most important use, though. The fact is that a jump message transfers control to whatever it jumps to, making it suitable for a number of different situations, such as:

- reassigning focus,
- beginning a ‘subroutine’,
- a dummy operation to execute parser statements.

In order to actually jump to another component (which includes a page), merely select the component in the list which opens once you select this message. You can jump to any component, although the reaction due to the jump might differ.

If you jump to a page (not a component on the page), the jumped to page will display itself. If you have set any effects on the page, these will take affect. All components on the page will receive a message to paint themselves and to evaluate any

parser statement associated with them. The paint sequence corresponds to the order in which they were created, or more precisely the order in which they have been layered (foreground / background). They will be painted from background to foreground. The component farthest in background, capable of receiving focus, will receive it.

If you jump to a component on a different page, then the page will be displayed, the components will be painted in the order as described above, and any parser statements will be evaluated. The difference is that the default focus assignment is overridden by the component jump. If the component the jump is directed to is capable of receiving a focus, it and not the component farthest in background, will receive focus.

Special consideration must be given if the component that receives the jump has been set to be invisible. In this case the page will not be displayed. Let's look at an example. If you have placed a rectangle on page two, along with a number of other components and you jump from page one to the invisible rectangle, then all links associated with the rectangle (Goal of a jump), as well as the associated parser statements will be executed. You can use this feature to launch activities without letting the normal effects of a jump become visible.

If you jump to a component on the same page, more or less the same process takes place. The page is not redrawn though, unless some other link is executed that invokes a repaint. Again, if you jump to an invisible component, the jump and all links associated with the target component will be executed. If the component could receive focus, but has been set to be invisible, then it will not accept the focus and focus will remain where it originally was (again, assuming that no link is activated that otherwise changes focus).

A jump message is also used to initiate a subroutine. This is done by dropping a placeholder component (most commonly a rectangle is used) on the desktop. Next, all desired links are placed on the rectangle. The event they trigger on though is Goal of a jump. When the rectangle is jumped to, then it is actually receiving a Goal of a jump event. Lets view this process in more detail:

Sending Component	Receiving Component	Description
Left mouse button released → Call to...		The component transfers control to the receiving component.
	Left mouse button released → Change Attributes Color to Green	This link is not executed, since the control has been received not via user interaction on the component, but via transfer from another component.
	Goal of a jump → Sound	This link is processed, since control has been received through the jump message/event. The sound message is executed.
Next link'		Control is back at the initial component.

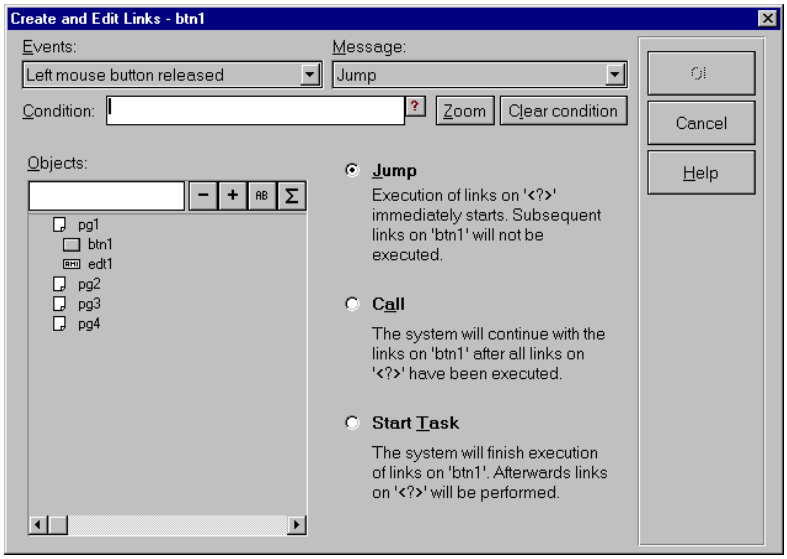
It must be noted, that the behavior of the sending component during the period in which the receiving component is processing its links, can be influenced by the setting of the switches on the right of the dialog box.

mindmap also supports computed jumps. When you select a component in the component list, mindmap will use the component as an absolute place to jump to. There might be situations in an application you are building, in which you would like to jump to a page or component which is determined at run time. To enable a computed (or parsed) jump, you must point the jump to a component which contains the destination. In this case the jump doesn't actually jump to the specified component, it uses the contents of the component as a pointer to the 'real' destination.

Let's use a little example to clarify this feature. Place an input field on Page1 of the application. Place a command button next to it. Now create a second and a third page. Go back to the first page and place a link on the command button which reads:

```
Left mouse button released   Jump to edt1
```

At the top of the section displaying the component list in the link dialog box, you can see a row of buttons:



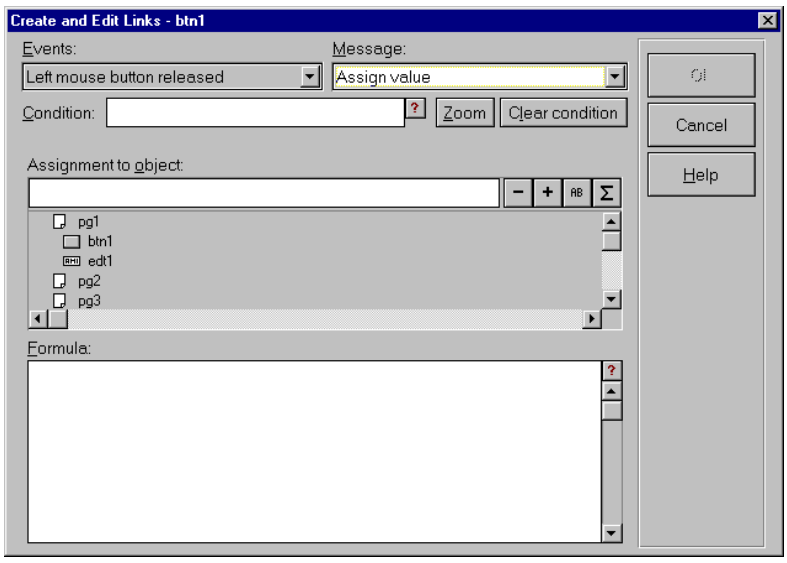
Chose a component as the target of the jump

Next to the plus (‘+’) and the minus signs (‘-‘) you will see two additional buttons. The button labeled ‘AB’ toggles the component list to display in alphabetical order. The button with the capital sigma (‘Σ’) is the one we need in this context. If you selected the input field labeled **edt1**, this name should appear in the area to the left of the four buttons. If you now click on the Σ-button, the list of components will be shaded and the component’s name listed in the field will be unshaded. The Σ-button will remained pressed. If you now leave the dialog box (by clicking the OK button) and return to the list of defined links, you will notice that the component’s name has been placed in square brackets (‘[ ]’). This symbolizes that the component, in this case **edt1**, is not being used as a jump target, but as a supplier of the pointer.

Assign Value

Once you have selected this message, the dialog box will change to reflect the options associated with this

message:



This dialog assigns a computed value to a component

This is the method by which a specific value can be assigned to another component. If you employ the parser, then you can assign a statement to a component, which will be evaluated when the component is instructed to do so. The general conceptual syntax for the Assign value statement is

```
component x ← statement
```

with component x being the component selected in the upper section of the dialog box. (Please note that only those components that are capable of having a value assigned to them are displayed in the list). The statement belongs in the area beneath the list of components. The statement itself must adhere to a certain syntax.

➤ The valid syntax of statements is described extensively in conjunction with the parser on page 388.

A statement may include:

- constants or values
- operators

- references to other components
- functions

Thus, a valid statement might be:

```
sqrt(component x) * (10*pi)
```

or

```
substr(component y,2,3)
```

The first statement would assign the result of the mathematical operation on the component-x as described in the statement. The second statement would assign the result of the string operation to the appropriate component.

Please note that you do not include any assignment or equality operators such as '=' or '<' in the statement. By selecting the component in the list, you are already making the assignment and you must merely supply the rest of the 'equation'.

The component to which you assign the value, does not necessarily have to be able to deal with the assigned value. You can use a component as a storage location for values. Often, rectangles are used as temporary storage locations for values, much as they are used for subroutines.

Let's look at a little example of this concept:

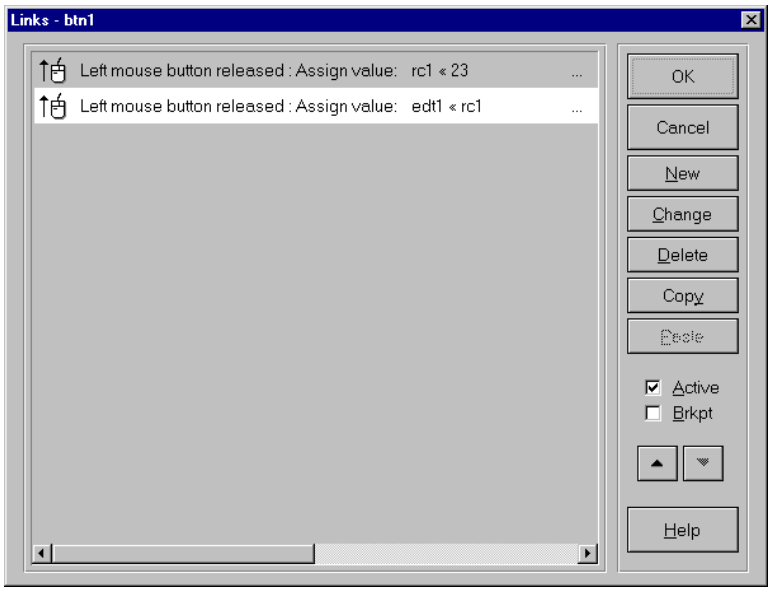
Place a command button, an input field and a rectangle, on the desktop. Select the command button and put the following link on it:

```
Left mouse button released- Assign Value 23 to rc1.
```

Next, define the following link on the same command button:

```
Left mouse button released- Assign Value rc1 to edt1.
```

Note that this assumes the rectangle has been given the name rc1 and the input field has been set to edt1.



Two links on btn1 assign values to different components

Please note that this little example would not function properly, if you were to reverse the order of the two links. The links are processed by mindmap in a top-to-bottom order.

The first link assigns the value to the rectangle and, even though the rectangle can't deal with the assignment in the conventional manner, it does store it. The second link takes the value of the rectangle and displays it, since the edit field knows how to deal with values.

Drag&Drop

mindmap's mechanism of drag&drop is commonly used to move information from one component to another. While the assign value command is designed to evaluate formulas and store the result into a component's formula attribute, drag&drop can be thought of a mechanism which moves larger amounts of information, and especially types of information that cannot be calculated with, such as bitmaps.

Please also keep in mind that the drag&drop facility is used in conjunction with mindmap's encapsulation feature (client/server), which makes it possible to move information between components that reside on two different computers.

Drag&drop is accomplished using only a few rules. A component is either capable of:

- sending data,
- receiving data or
- both.

Information from a component can be offered in three different ways:

Data Type	Description
ASCII	Text information, sometimes limited in size according to a component's limitations. Text can include multiple lines, separated by the carriage return-line feed sequence or multiple columns within the same line can be separated by the tab character.
BITMAP	Graphical information, usually device independent, i.e. rendering of an image occurs in the most suitable way with respect to the computer's graphic adapter.
METAFILE	Also graphical, but vector-oriented, data.

All three data types are internally stored in the same format as used for the Windows clipboard.

Obviously, each component renders its information as a suitable type. Some components can even supply their data in multiple formats: a graphic component, for instance, can supply its contents as an image (either BITMAP or METAFILE), or it can send the image's file name, if requested by another component.

During the drag&drop exchange, both components negotiate a data type which both can support. Thinking of a drag&drop operation being established between graphic component and an input field, the graphic component will first offer its data as an image. Since an input field cannot deal with bitmaps, this offer will be refused, which in turn causes the graphic component to send the image's initial file name to the input field. If this negotiation also fails, nothing will be transferred, both components appear to be incompatible.

Not every component is capable of sending and/or receiving drag&drop information. The following list contains all components that participate in the drag&drop negotiation. Also, the data types that can be supplied or accepted are listed:

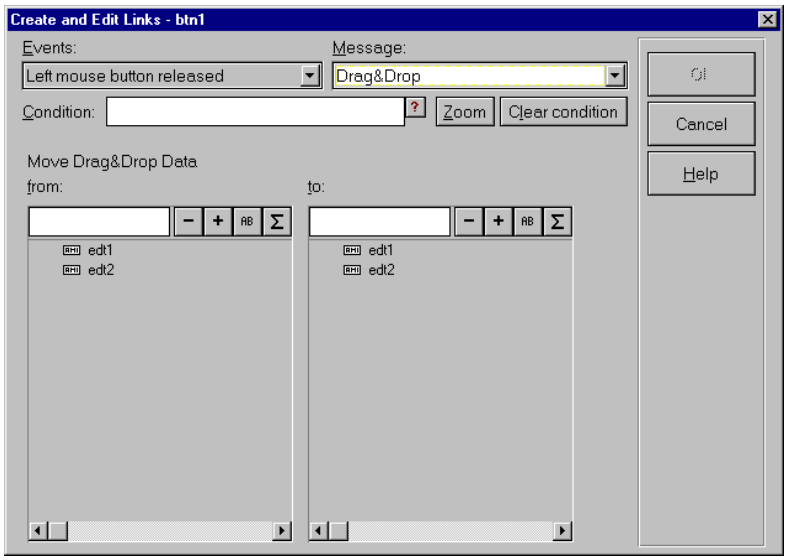
Component	Sends	Accepts
Graphic component	Bitmap or MetaFile, File name, if known, as text. The graphic component does not know about a file name if the image has been copied from the clipboard.	Bitmap or MetaFile. Valid file name, i.e. complete directory, file name and extension information. According to the extension, the graphic component will choose a graphic input filter to load the image.
Input field	Text, limited by 32KB. If the component is a multiline input field, the lines will be separated by the carriage return-line feed sequence. Data classes other than string will be converted into strings before they are transferred. An input field will always make it content available in the specified format (date, currency, etc.), as well as a string.	Text. Large text data will be truncated to the limitation of the input field. Should the input field be formatted to a class other than string, the information is expected to be a string and will be translated appropriately. An input field will attempt to interpret the incoming data in its specified format (currency, date, etc.). If this fails, it will interpret it as a string.
List box	Text (file attribute not set): Depending on the settings in the component's drag&drop settings, either the highlighted row(s) or all rows are sent. Rows are	Text (file attribute not set): The received text is appended to the end of the list. If this text contains multiple lines, multiple rows will be created.

	separated by the carriage return-line feed sequence. Text (file attribute set): The contents represents file names and will therefore be sent as such, including the full drive and path specification. Graphic (Bitmap only): If the list box is set to Show Bitmaps, the currently selected bitmap will be sent, along with the other representations of the data. Collapsed list box: The selected row will be stripped of its leading indentation marks ('\'), before it is sent.	Text (file attribute set): The incoming text will be interpreted as a file name and displayed accordingly, including the full drive and path specification. Graphic (Bitmap only): If the list box is set to Show Bitmaps, a valid file name is expected. The list box will attempt to load the graphic file and stretch it to the width of the list box (keeping the ratio constant). Collapsed list box: The first characters will be interpreted as indentation markers. Each '\' will cause the following string to be indented by one column.
Data table	Text: Multiple cells in the same row will be separated by the tab character. Multiple lines will be separated by the carriage return-line feed sequence. If a specific column has been selected in the drag&drop option dialog, only the contents of this column will be sent, separated by crlf. The data table component either sends its entire content or only the highlighted rows, depending on the settings in the	Text: To be copied to multiple cells of the data table, incoming text data is expected to contain multiple rows, separated by crlf and/or columns, separated by the tab character. If multicolumn text information is received into a data table having selected a specific column in the drag&drop dialog will result in the data copied into subsequent columns beginning with the specified column.

	drag&drop configuration dialog.	New rows will automatically be created if necessary.
Input/Output - clipboard	The component will send either text, bitmap or MetaFile data, depending on the current contents of the clipboard. If the clipboard is empty, nothing is sent.	The component will accept all types of information and copy it to the clipboard, making it available for other programs.
Input/Output - file	The component will send either text, bitmap or MetaFile data, depending on the contents of the selected file. If the file format cannot be determined, its contents will be sent as text.	The component will accept all types of information and create a file or append to an existing file if text data was received. Make sure that a file extension suitable for the type of data has been specified. If bitmap information has been received, the component will always create a device independent bitmap file (*.DIB or *.BMP).
Input/Output - printer	The component cannot send any information.	Text: Text data will be printed on the specified printer using the printer's default font. The output is limited by a single page, regardless of the amount of data being received. Graphic: The printer will print the graphic. The size of the output can be modified using the component specific

		dialog.
Output page	The output page will send a rendering of the components on its page.	The component will either accept text or graphic and will create an appropriate component on its output page.
MCI component	The component cannot send any information.	Valid file name, i.e. complete directory, file name and extension information. According to the extension, the MCI component will attempt to open the media device. (same functionality as Open File link command.

Once you select the drag&drop message, the dialog box will change its appearance. On the lower left-hand side of the dialog box, a list of components capable of supplying data via drag&drop is displayed. Immediately to the right, you will see a list of components in the application which have the ability to react to drag&drop data.



Select a sending and receiving component for the drag&drop mechanism

Let's use a simple example to explain how this message works. Place a button and two input fields on page 1. Select the command button and place a link on it. Use Left mouse button released as the event and select the Drag&Drop message. The two list boxes will only contain components capable of sending or receiving data via this message. Therefore, the button you have placed on the screen will not appear. Select edt1 as the component from which the data is requested and select edt2 as the receiving component. Put mindmap in run mode and type some data into the input field edt1. Now, click on the command button and the data should be copied from edt1 to edt2.

Let's take this one step further. Return to edit mode and click on the icon representing the import of graphic files. Select BMP as the format and search your system for a .BMP file. Generally, your Windows directory will contain such files. While you're at it, try to remember a second file name (or better yet, take a note of it). Accept the selection and place the bitmap on the desktop. Next, select the bitmap and set its attribute to drag&drop enable. (This is the icon with the little hand dropping the ball). You should still have a command button and an

input field placed on the desktop. If not, place one of each on the page. Select the command button and place a link on it. As an event, use the omnipresent Left mouse button released, and, as the message, use drag&drop. Now select the input field as the originating data source and select the bitmap as the receiving component. Accept the link, go back to the desktop and put mindmap into run mode. Your screen should look something like this:

A graphic component has received a file name through drag&drop. It loads an image file.

Select the input field and key in the name of the file you either memorized or have written down. Please make sure that you type in the complete path for the file. When you click on the command button, a drag&drop operation will be performed. Strangely enough though, the existing bitmap will be replaced by the bitmap contained in the file you have specified in the input field.

What has happened? The bitmap has received a file name as data via the drag&drop message. It has then attempted to display the file name, which it can't. It then attempted to locate a file by that name (and hopefully succeeded), has opened the file, read it, and displayed the contents in place of the existing

bitmap. The existing bitmap has taken on the roll of a placeholder.

Reverse the order of sending and receiving in the above example - drag&drop the bitmap to the input field (remember to set the attribute of the input field to enable incoming data via drag&drop). Go into run mode and execute the link. Interestingly enough, what will happen is that the bitmap will display its file name in the input field. What has happened now, is that the bitmap started out by sending the actual bitmap to the edit field. It responded by informing the sender, that it could not deal with this form of data, only with alphanumeric information. The bitmap component then sent it a file name. This the input field could deal with and immediately displayed it.

Try the same procedure with a list box. Place a list box on the desktop, along with a button and a bitmap. Set the drag&drop attribute of the list box to receive data. Instruct the button to perform a link, whereby it is to send data via drag&drop from the bitmap to the list box. Put mindmap into run mode and execute the link. As presumably expected, the list box displays the file name associated with the bitmap. Now go back into edit mode and set the specific attributes of the list box to 'Show Bitmaps' (bottom left hand corner of the specific attributes dialog box). Now get back into run mode and execute the link. Well, the list box now displays the bitmap you sent it. If you execute the link multiple times, the list box will display multiple instances of the bitmap.

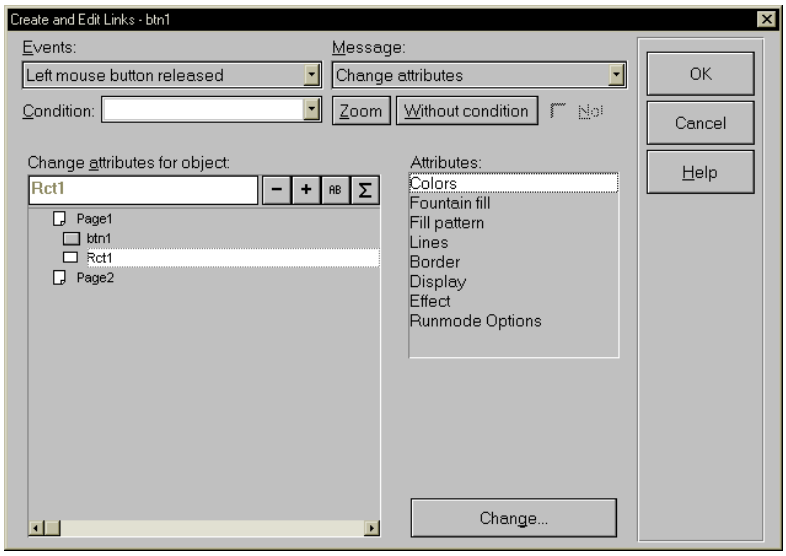
The drag&drop message also supports the computed drag&drop feature. When specifying the sending component, click on the Σ-button after selecting the component. Keep in mind though, that the supplying component must be capable of containing a pointer (i.e. a graphical primitive cannot be used here). Then select the destination. When the link is executed, it will go to the specified component, pick up the pointer, and get the contents of the component to which the pointer is pointing. The contents will then be forwarded to the receiving component.

The same procedure can be used the other way around. The receiving component can contain a pointer to the component which is really supposed to receive the data. This way you can pick up data from a component and send it right back to itself.

Change Attributes

During the course of an application, it often becomes necessary to change one or many attributes of a component. You might wish to change the color of a graphical primitive to reflect a change in a certain status, or you might want to gray the text on a button and simultaneously switch it to be inactive.

To achieve this, you merely have to define the event which will trigger the change, select the component you want to perform the attribute change on, the attribute to be changed, and the new value it is to take on:



This link will modify an attribute of a component

All attributes that can be changed in edit mode, can also be changed in run mode. Obviously, this will always be at the discretion of the application developer, but, nonetheless, all attributes can be changed.

If a component registers its own attributes, these are accessible via the entry Object in the attribute list.

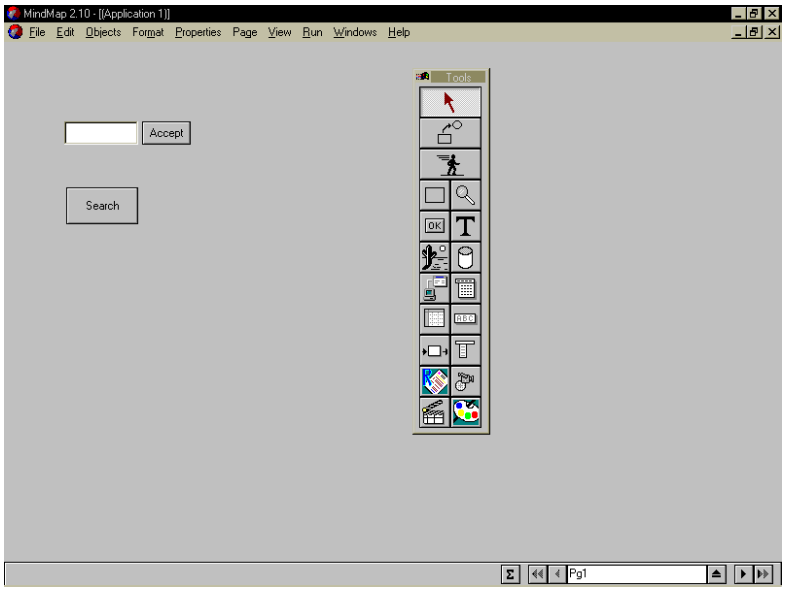
Let us use two examples to demonstrate how this message is used.

First, we will simply toggle the color of rectangle:

1. Place a command button and a rectangle.
2. Access the link facility on the command button.
3. The event should be Left mouse button released.
4. The message should be Change Attributes.
5. Select the rectangle in the list.
6. Double-click on the Colors attribute and select red.
7. Acknowledge the link.
8. Create a new link using the Right mouse button released as the event.
9. Proceed as described above, but select a different color this time.
10. Acknowledge the input and return back to the mindmap screen.
11. Put the application into run mode and toggle the color.

Now, let us create a somewhat more complex link, using the Change Attribute message.

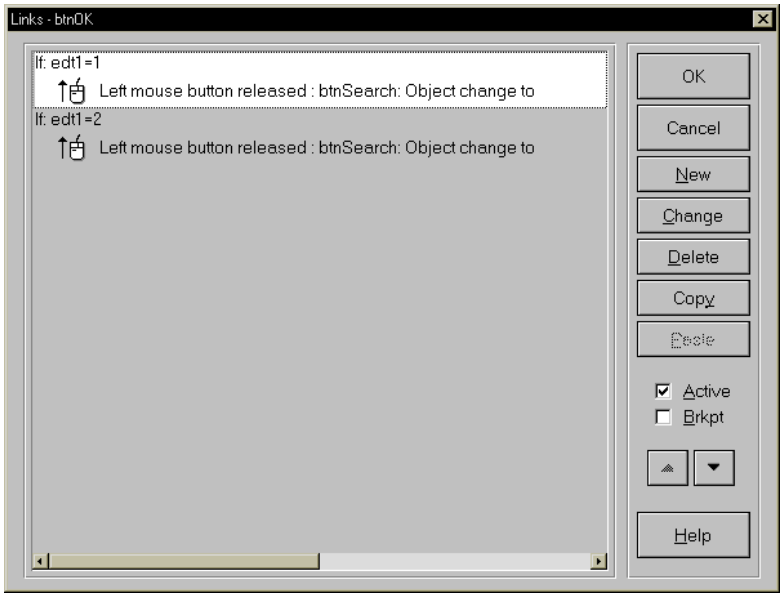
The goal is to turn a command button inactive, conditional on the contents of an input field:



A little application which changes an attribute of a component

1. Place two command buttons and an input field.
2. Label one command button with OK, label the other one with Search.
3. Name the first button, btnOK, the second one btnSearch.
4. Set the format of the input field to be Number.
5. Access the links on btnOK.
6. As the event use Left mouse button released.
7. As the message select Change Attributes.
8. Now select btnSearch and its Object attribute.
9. Check the check box labeled inactive.
10. Select the Conditional field just beneath the Event list and enter **`edt1=1.`**
11. Acknowledge the link.
12. Create a new link with the inactive check box unchecked and enter as a condition, **`edt1=2.`**

Your links should look like these:



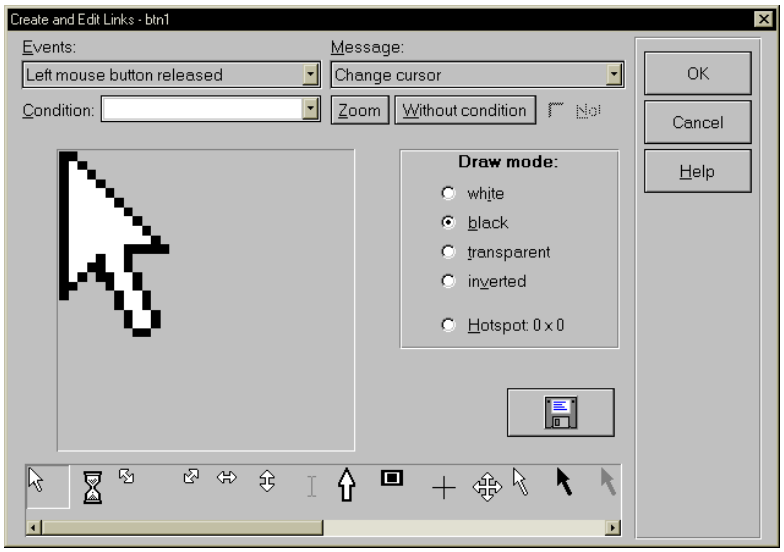
Two conditional links change the appearance of a button component

Put this little application into run mode and enter a 1 into the input field. Once you have released the left mouse button on the Accept button, the Search button should become inactive. Entering a 2 into the input field, and subsequently releasing the left mouse button on the Accept button, will switch it back to the active mode.

Change Cursor

This message is used when you want to notify the user of some, possibly non-obvious, condition in your application. Examples might be that you want to:

- cue the user as to the existence of a hot spot,
- signal that a special operation is taking place (printing, drag&drop, etc.)

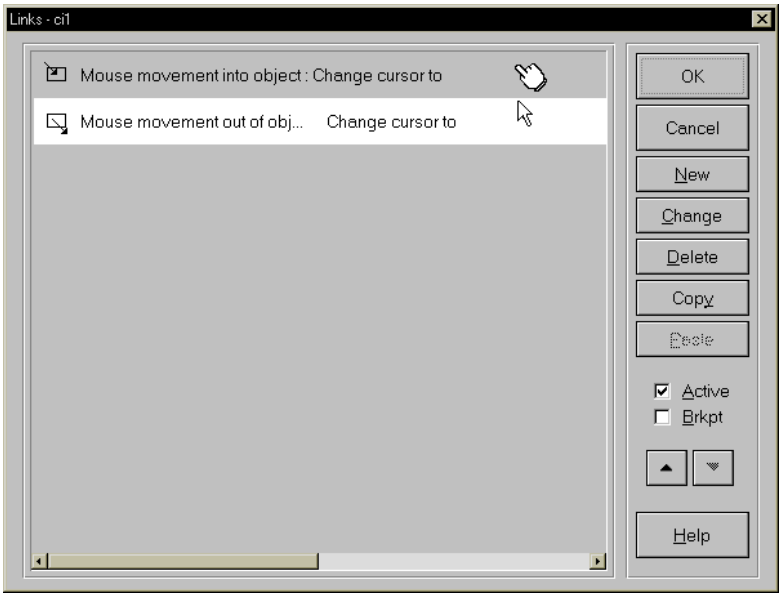


Select from several predefined mouse cursors or easily create your own

We will assume you want to change the cursor to indicate the location of a hot spot on a bitmap:

1. Place a bitmap on the screen.
2. Select a circle (graphical primitive) and place it on top of some significant area of the bitmap.
3. Select transparent as the fill color.
4. Select transparent as the line color.
5. Access the link facility on the circle.
6. Select Mouse Movement into Object as the event.
7. Select Change Cursor as the message.
8. Choose one of the cursors offered in the scrollable list.
9. Acknowledge the link.
10. Create a new link with Mouse Movement out of Object as the event and the same message.
11. Now, pick a different cursor from the list.
12. Acknowledge this link.

Your two links should look something like these:

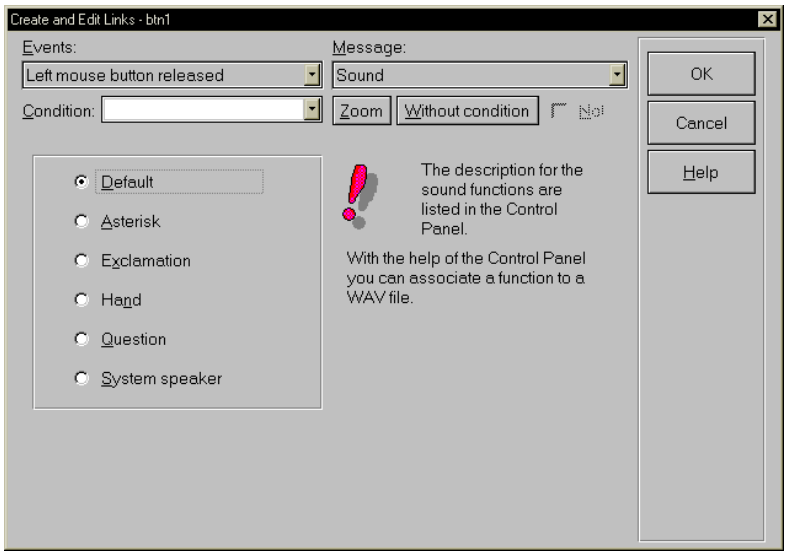


The mouse cursor will be changed when the mouse moves over a component

Keep in mind that you must always change the appearance of the cursor back to its initial state, dependent on some valid event, even if you jump to another page.

Sound

This message is used to give the user some simple form of audible feedback, depending on some action. It will function on all Windows systems which have a loud-speaker. You should not confuse this message with the multi-media message MCI command and Multimedia. This message is the least common denominator on all Windows systems. Therefore, it is actually only capable of generating simple sounds, such as beeps.



Create different sound effects

This message picks up the valid settings on your system that have been assigned in the Windows control panel. Please note that your setting does not necessarily match those on other systems. Please keep this in mind when planning to deploy your applications to other systems.

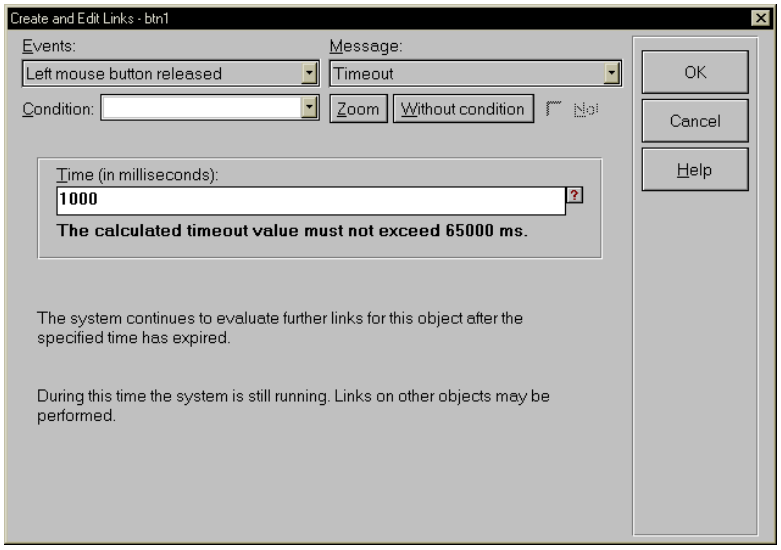
Building a link based on this message is extremely straight forward:

1. Place a component and access its links.
2. Select an event and pick the Sound message.
3. Click on one of the aforementioned sound options in the selection list.
4. Acknowledge the link and put the little application into run mode.
5. Generate the event and you should hear the sound.

Time-Out

This message does exactly as the name implies — it times out for a specified period. During this period, all links on the component which starts the time-out are

not processed. Any other links on other components, which may become active during the time-out, will be processed.



This link pauses the execution of links for a given time

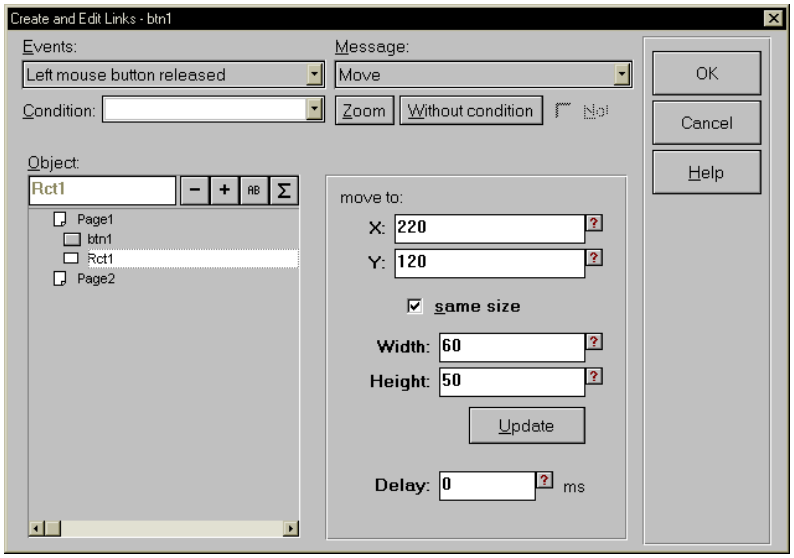
The time specified in the input field is defined in milliseconds (1 sec = 1000 msec). This field is also evaluated by the parser at run time, so that a statement can be input, instead of a constant. Thus, the actual time-out period can be kept variable.

Move

This message allows certain components to be moved at run time.

The movement is limited to the page on which the component has been placed. Moving it outside the coordinate space of the page, will make it inaccessible for the user. Returning it into the visible area requires the execution of an additional Move message.

All components can be moved during run time, although in some cases, it is hard to conceive of a reason to do so.



This link moves a component in run mode

Again, let us build a simple example:

1. Place a command button and a rectangle.
2. Access the link feature on the command button.
3. As an event use Left mouse button released, as a message use Move.
4. Select the rectangle from the list of components below the event list.
5. Click on the Update button on the right side of the dialog box (This should cause the x- and y-coordinates of the rectangle to appear in the corresponding fields).
6. Retain the displayed Width and Height and enter a Delay of 1000 msec.
7. Acknowledge the entries and return to the mindmap screen.

Since all five fields are parsed at run time, you can also enter a computed statement.

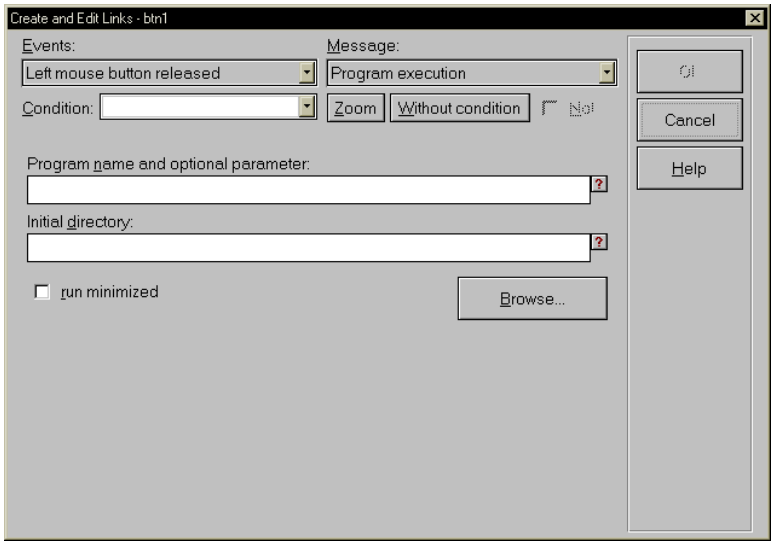
While still in edit mode, move the rectangle to a different location on the screen. Put the little sample application into run mode and click on the command button. With the delay of a second, the rectangle should return to the position at which it was prior to you moving it manually.

The Update button picks up the x- and y-coordinates, as well as the size of the selected component. Obviously, you can enter any other (valid) value. By default, the size of the selected component is retained.

When the component is moved with a delay, then a frame with the size of the component is drawn and it is moved with the specified delay to the designated screen position. It is not possible to move the component as it is seen in its static form.

Program execution

This message is used to launch other executable files (or programs) from within mindmap. Keep in mind that launching an application in Windows implies that another process is started. This means that control will immediately return to mindmap -- links following the program execution will immediately be executed.



This link launches another application

Let's try a little example. To launch the MS Windows calculator from within mindmap, perform the following steps:

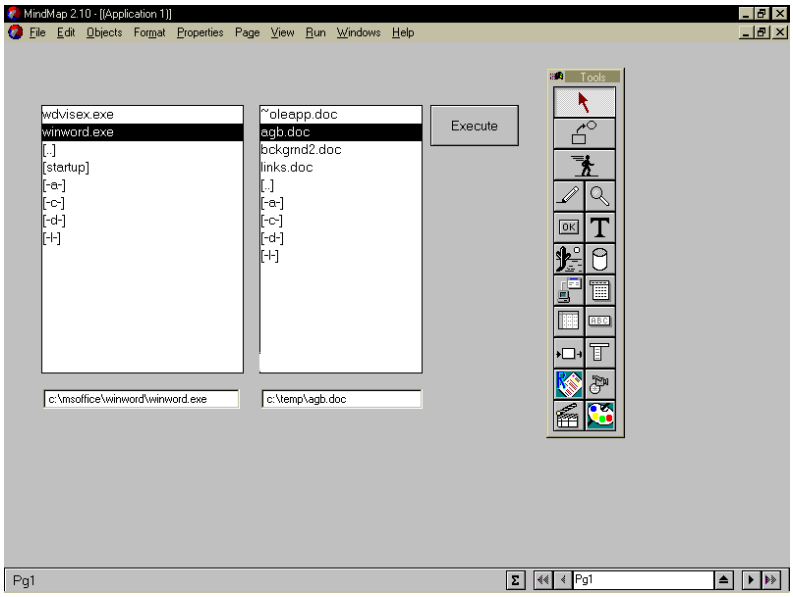
1. Place a command button.
2. Access the link feature on the command button.
3. Use Left mouse button released as the event and Program Execution as the message.
4. Click on the Browse button and navigate the file list until you find the calculator (under normal conditions, it will be located in your Windows directory).
5. Acknowledge the link and return to the mindmap screen.

You can use the parser function GetModuleHandle to verify if an application has already been loaded (see page 420 for a description of this function).

Place the little application into run mode and click on the command button. mindmap will pass control to Windows, which in turn will locate the Windows calculator and launch it. If you minimize the calculator, return to the little application and click on the command button again, another instance of the calculator will be launched. To avoid launching multiple instances, you must terminate the application you have started, before you attempt to launch it again.

Both fields in the dialog box are parsed, meaning you can also enter any valid statement. This is especially useful if you wish to launch the application in conjunction with a specific file. If you wish to launch a word processor, together with a document the user selected from a list, then you simply use the string concatenation feature to add the selected file to the program name.

Let's construct another little example. We want the user to pick the word processor application from one list of files and the document they want to work on from the second list. Clicking on the command button should launch the application and pass the document file, via the command line interface.



Here's an application that starts a program and automatically opens a document

Here are the necessary steps:

1. Place two list boxes. Name one ProgramFiles and the other DocFiles. Set the component specific attribute of both list boxes to display files with full path display.
2. Place an input field beneath each list box. Name the one beneath the ProgramFiles list box ProgramFileName. Name the other one DocFileName. (In a real application both input fields would be set to be invisible at run time).
3. Place a command button and label it with a caption, such as Execute.
4. Select the list box ProgramFiles and access the link facility. As an event, use Application Started. As the message, use Combo-/List box. Here, select the option, Directory, and enter the string "C:*.EXE", including the quotation marks. This will only display files with the extension .EXE.
5. Place another link on the list box. Use Double-click as the event and Drag&Drop as the message. Drag&drop the

contents of this list box to the input field ProgramFileName. (This will assure that the full path is used).

6. Do the same for the other list box, with the exception that the string you should enter is "C:*.DOC". This will display only those files with the extension .DOC.
7. Also place a link to drag&drop the contents of DocFiles to the input field named DocFileName.
8. Select the command button and access its link facility. Use Left mouse button released as the event and Program Execution as the message.

In some cases, it is necessary to load an application, but to keep it minimized. To accomplish this, mark the check box labeled *Run minimized*. If you launch an application using this switch, then the application will not appear in foreground, but it will be running.

9. Enter into the field labeled Program name... the following string
   ```
   ProgramFileName + " " + DocFileName
   ```
   The quotations marks enclose a blank which is concatenated between the program name and the doc file, since this is the convention by which files are specified in the command line (an old MS-DOS convention).
10. Return to the mindmap screen and place this little application into run mode. The left list box should fill itself with the top-level directories and the drives defined on your system. Navigate the structure until you find the program you wish to use. Every time you double-click on an entry in the list box, it will appear in the input field beneath the list box.
11. Navigate the right list box until you have located the document file you wish to work on. Here, again, every double-click will pass the selection from the list box to the input field.

Once you have located the program file and the document file, click on the command button.

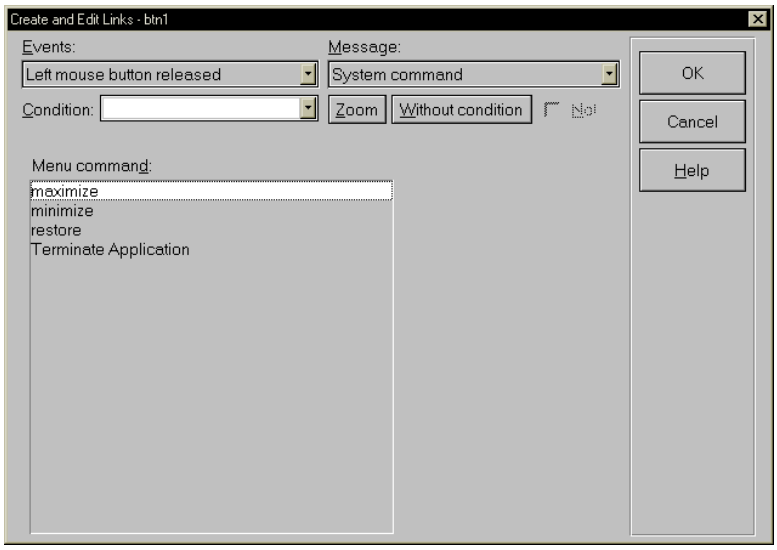
System command

mindmap offers a set of commands which affect itself.

- Maximize: This option allows you to let the user expand the application to full screen.
- Minimize: Conversely, this option allows you to let the user minimize the application (iconize). If you have used the feature which allows you to define your own application

icons, this will be displayed, once the application has been minimized.

- Restore: Causes a mindmap application, that has been minimized, to restore itself to its initial size and position.
- Terminate Application: Executing this message will terminate the mindmap application and return control to the Windows environment. If the application you are executing is not running as an EXE file, then this message will function as a toggle back into the edit mode of mindmap.



A link can change the appearance of an application's window

MCI Command

Windows offers a large set of multimedia extensions which can control audio/visual devices like sound boards, CD players, Video Disc players as well as the Video for Windows system.

All of these devices are controlled through a common interface. The basic approach to this multimedia control interface (MCI) is a command language. Windows itself defines only a minimum set of commands. Various types of multimedia devices

may introduce new commands to this syntax to cover their special features.

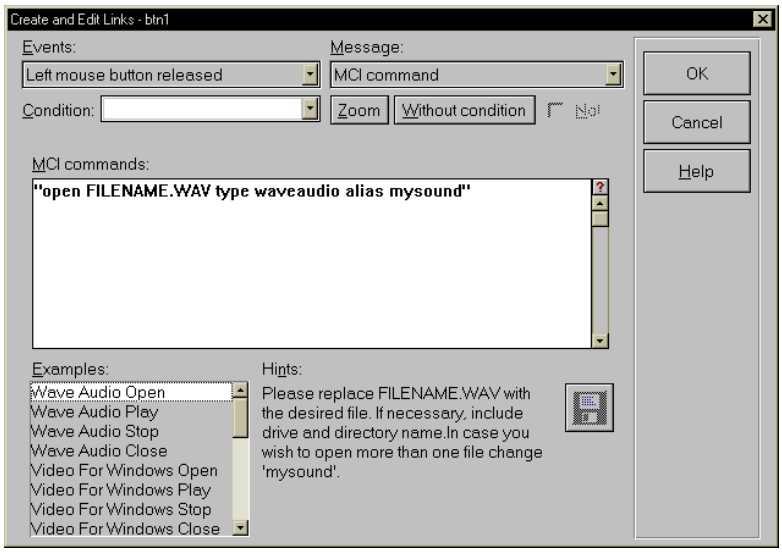
mindmap offers a convenient interface to this multimedia command language. The benefit of this approach is that low level calls to the device interface are possible. The disadvantage is that you will have to concentrate on the various options of a particular device by reading through the device's documentation.

The link dialog contains a generic set of predefined instructions you can chose from to build your MCI commands.

Please note at this point that a device is always identified by its type name. Valid type names may be:

MCI Type	Description
waveaudio	Windows sound through a built-in sound board.
avivideo	Video for Windows if running under or if it has been properly installed as a Windows extension.
cdaudio	a CD-ROM drive as an audio CD-Player.

Once an MCI device has been opened, it is referenced through an alias name in any further commands that apply to the open device. This device name is arbitrary. However, you should select a suitable name to make it easier to associate a particular statement with the type of device it is applied to.



Execute command strings on the MCI interface and control various MCI devices

If the device deals with files (Video for Windows and Wave, but not CD Audio), then the desired file name has to be included.

If you pick various open commands from the pick list at the bottom of the link dialog, you will find a FILENAME with the most common extension for this type of media. By pressing the button with the disk symbol, you may select an appropriate file from a standard file dialog box.

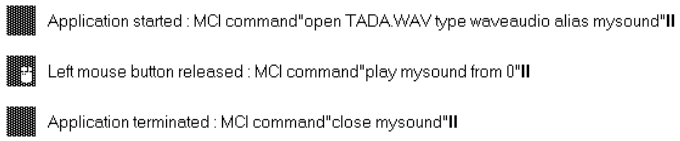
Since all MCI commands are parsed, you may freely construct your own MCI command strings from the contents of other mindmap components, such as input fields or list boxes.

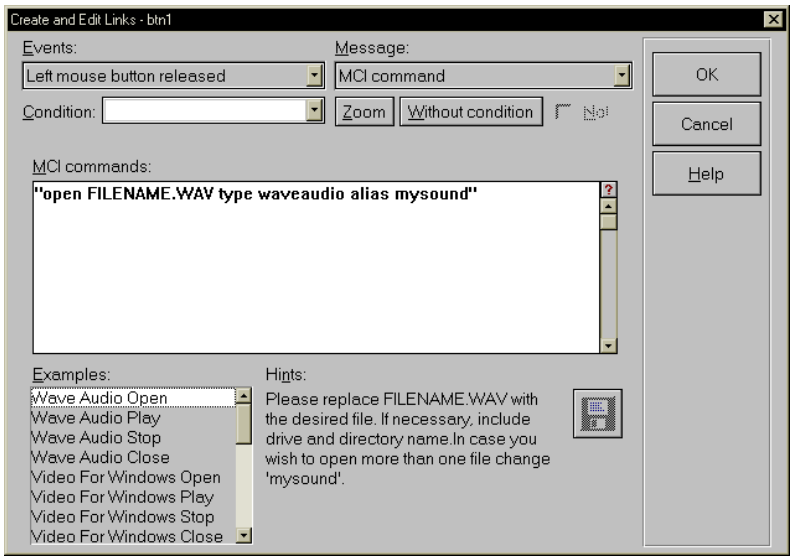
The following example will open and play a Windows wave file. Execution of subsequent links will pause, until the file has been played.

```
"open C:\\WINDOWS\\TADA.WAV type waveaudio alias
mysound"
"play mysound from 0 wait"
"close  mysound"
```

If you omit the word wait, mindmap will play the file and continue to execute subsequent links. In this case make sure that the sequence of commands is split across different events. Playing the sound file would immediately stop, if the MCI command close mysound would be executed immediately after the command play mysound from start.

Try to put the following sequence of MCI commands on a button, given that the wave file exists:





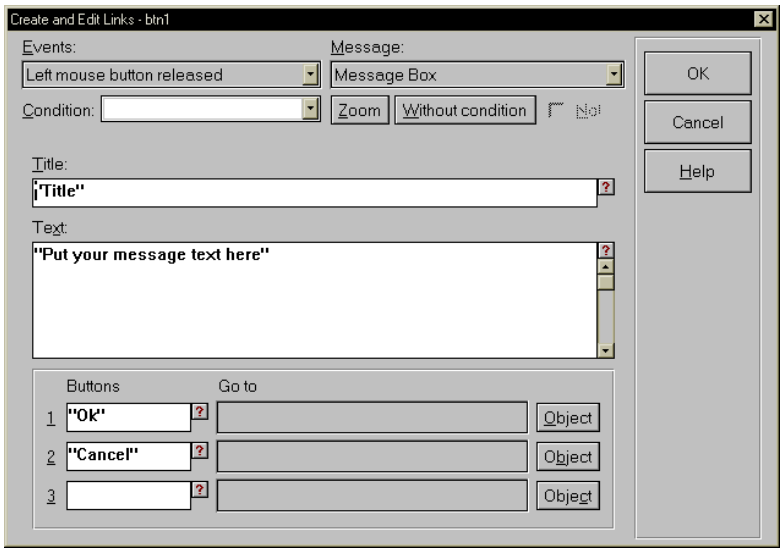
This link opens a sound file to be played on your sound card

➤ Another way to access MCI functionality is through the MCI component, described on page 221.

Message Box

This message allows you to post a message box with various alternative buttons. This feature is generally used to inform the user of a certain condition, to offer options regarding the general flow of the application, or to post an error message.

Place a button on the mindmap screen and access its links. As an event, use Left mouse button released and select the Message box as the message. The following dialog box will be presented:



Fill out this form to prompt the user with a message box

The first entry you can make relates to the title or caption of the message box you wish to display. mindmap has put a placeholder into the field called Title. Please enter the desired caption into this field and make sure that you enclose the string in double quotation marks. Since this field is parsed, not enclosing a string in quotations marks will have the parser attempt to interpret the string as the name of a component which contains the desired string. The corollary is that you can also enter a component name containing the string you wish to use as a title.

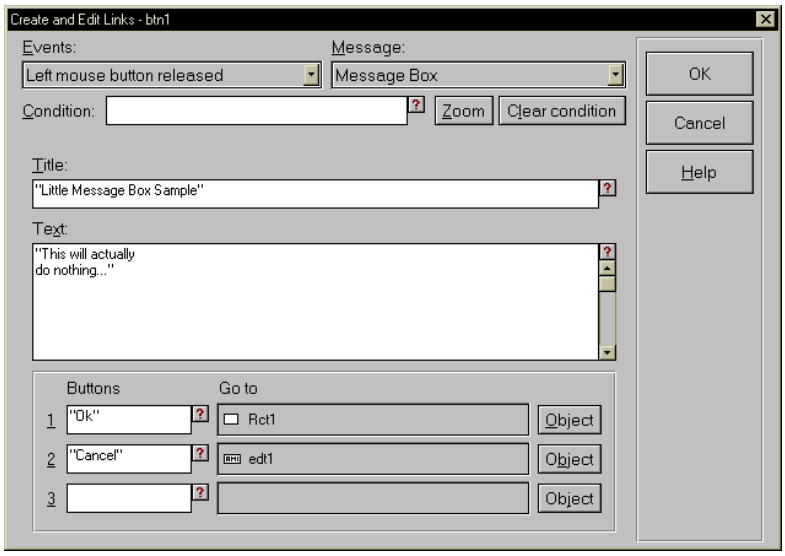
The next field is used to enter the actual body of the message text you wish to display. Again, this field is parsed. Thus, if you wish to enter a string which is to be displayed, then enclose it in double quotation marks. If, on the other hand, you wish to place a pointer (another component name) into this field, then do not enclose it in quotation marks. Please note that mindmap will automatically attempt to format the string, should it not fit into the boundaries of the message box. If you wish to force a new line (carriage return/line feed), then enter an **ENTER** or a **CTRL+ENTER**. In this case, the link will display a double vertical line, which depicts the position of the new line.

At the bottom of the dialog box you will find three fields which correspond to a maximum of three buttons you can display on the message box at run time. Enter the labels in the corresponding fields.

To the immediate right of each of the label fields, you will find a field titled Go To. This is where you can enter the location the button should jump to. You can either manually enter the component name to which you want to divert control, or you can click on the button labeled Object. This will pop up the list of all components.

Let's construct a little example. This example will display a message box with a caption and a body, along with two buttons pointing to other components:

1. Place a command button, an input field, and a rectangle.
2. Label the button.
3. Now access the link facility of the button, using Left mouse button released as the event and Message Box as the message.
4. Enter in a caption and a body for the message box.
5. Keep the OK label and point it at the rectangle (rc1).
6. Also retain the Cancel label and point it at the edit field.
7. Acknowledge the dialog and return to the desktop.



Inform the user about a certain condition of your program

Now let's deal with the components you are pointing to:

1. Select the rectangle and access its links.
2. As the event select Goal of a jump and as its message generate a Sound.
3. Acknowledge and return to the desktop.

Start your application. As soon as you click on the command button, the message box should appear. If you click on the OK button, the message box should disappear and you should hear a sound. Click on the command button again, this time click on the Cancel button. Again, the message box will disappear, but the focus will now be in the input field.

Component Specific Messages

Data Table

Since data tables are a standard mindmap component, this message will appear, even if you haven't placed a

data table in the application. You will not be able to execute a data table message though, until you have placed one in the application.

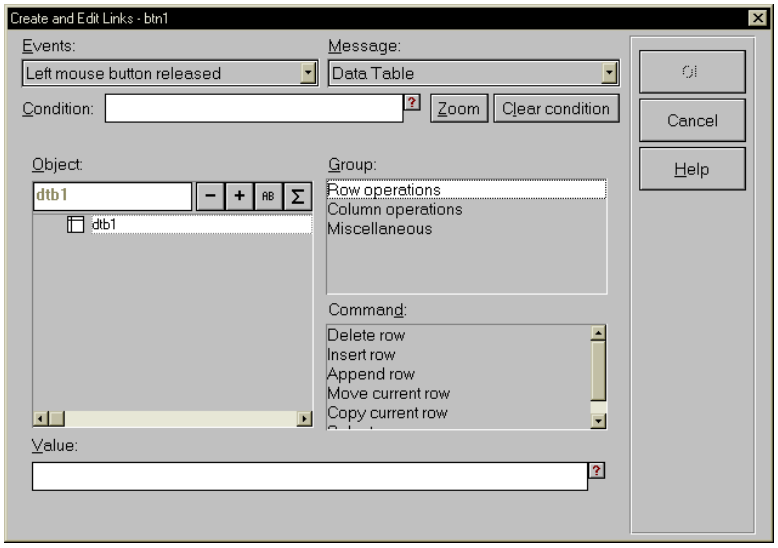
Once a data table is available, three classes of commands can be accessed. These are:

- Row operations
- Column operations
- Miscellaneous

As the labels suggest, the commands are either oriented at the row, the column or the table in general.

All row and column operations assume integer numbers. The data table message will automatically convert and/or round the values of strings, floating point, etc., to integers. Also note that row and column numbers start with 1.

Let's begin by looking at the row commands:



Select a function to change the appearance of a data table

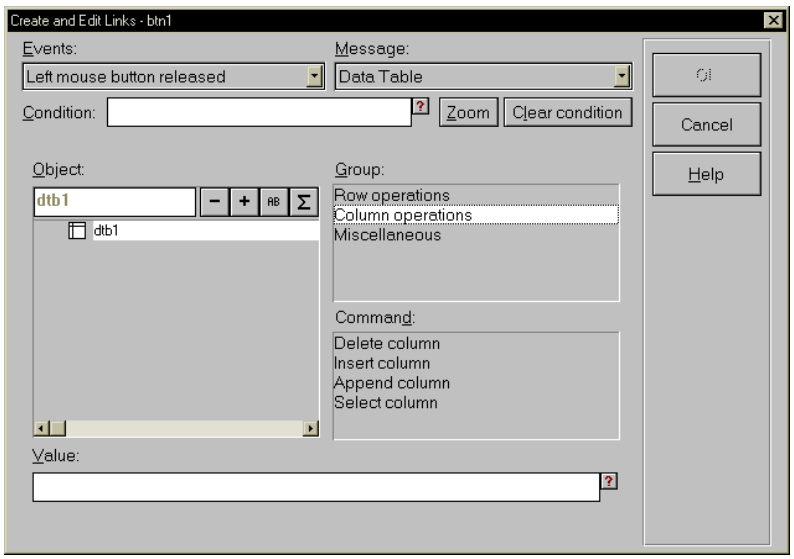
This group allows for six different commands:

Command	Description
Delete row	This command will delete the row specified in the Value section of the dialog box. Since this field is parsed, you may either enter an integer or a component containing an integer. mindmap

	always attempts to resolve a string entry as a component name (reference), before interpreting it as a literal. Therefore, you can cascade the pointer in the Value field; i.e., a component name, which contains a component name, etc. If the data table contains only one row, it cannot be deleted.
Insert row	This command will insert a new row immediately above the currently selected row. All rows - starting with and including the currently selected row - will be incremented by 1. The contents of the newly inserted row will be empty. The focus of the data table will remain with the previously selected row. This command will override the setting of maximum rows in the component specific attribute.
Append row	This command will append a new row to the bottom of the data table. The contents of the newly inserted row will be empty. The focus will remain in the selected row. This command will override the setting of maximum rows in the component specific attribute.
Move current row	This command will move the selected row to the new row specified in the Value field at the bottom of the dialog box. The moved row will be inserted into the new position. The old row from where the row was copied will be deleted. Since this field is parsed, you may either enter an integer or a component containing an integer. mindmap always attempts to resolve an string entry as a component name (reference), before interpreting it as a literal. Therefore, you can cascade the pointer in the Value field; i.e., a component name, which contains a component name, etc. If you do not specify a row number, the selected

	row will be deleted.
Copy current row	This command will copy the selected row to the new row specified in the Value field at the bottom of the dialog box. The copy of the row will be placed in a newly inserted row. Since this field is parsed, you may either enter an integer or a component containing an integer. mindmap always attempts to resolve an string entry as a component name (reference), before interpreting it as a literal. Therefore you can cascade the pointer in the Value field, i.e. a component name, which contains a component name, etc..
Select row	This command will select a row specified in the Value field at the bottom of the dialog box. Since this field is parsed, you may either enter an integer or a component containing an integer. mindmap always attempts to resolve an string entry as a component name (reference), before interpreting it as a literal. Therefore you can cascade the pointer in the Value field, i.e. a component name, which contains a component name, etc. If you specify a row number less than 1 or greater than the largest existing row number, nothing will happen. Any previously selected row will remain selected.

Now, let us review the column commands:



These functions deal with column manipulations of a data table

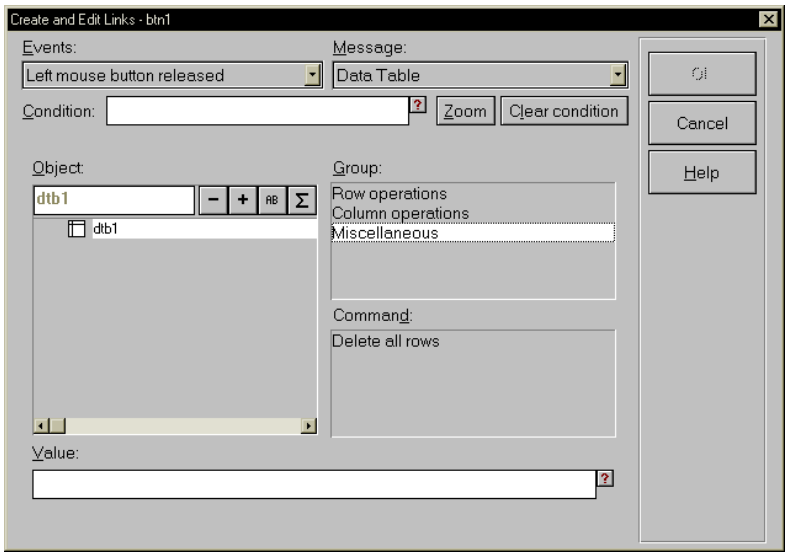
This group allows for four commands:

All column operations require that the switch '*Large Data*' be deactivated in the component specific attributes of the corresponding data table.

Command	Description
Delete column	This command will delete the column specified in the Value section of the dialog box. Since this field is parsed, you may either enter an integer or a component containing an integer. mindmap always attempts to resolve a string entry as a component name (reference), before interpreting it as a literal. Therefore, you can cascade the pointer in the Value field; i.e., a component name, which contains a component name, etc. If the data table contains only one column, it will be deleted. The consequence, though, is that the complete data table will also be deleted.
Insert column	This command will insert a column specified in the Value section of the dialog box. Since this field is parsed, you may either enter an integer

	or a component containing an integer. mindmap always attempts to resolve an string entry as a component name (reference), before interpreting it as a literal. Therefore, you can cascade the pointer in the Value field; i.e., a component name, which contains a component name, etc. The insertion will increment the column number of all columns greater than the column being inserted by 1. The column attributes of the inserted column will be picked up from the settings in the component specific attributes of the data table. Any column selections prior to the insertion will remain selected.
Append column	This command will append a new column to the right of the data table. The contents of the newly inserted column will be empty. The focus will remain in the selected row and column. This command will override the setting of maximum columns in the component specific attribute.
Select column	This command will select the column specified in the Value section of the dialog box. Since this field is parsed, you may either enter an integer or a component containing an integer. mindmap always attempts to resolve an string entry as a component name (reference), before interpreting it as a literal. Therefore, you can cascade the pointer in the Value field; i.e., a component name, which contains a component name, etc. mindmap will not highlight the currently selected column.

Now, we will look at the last group:



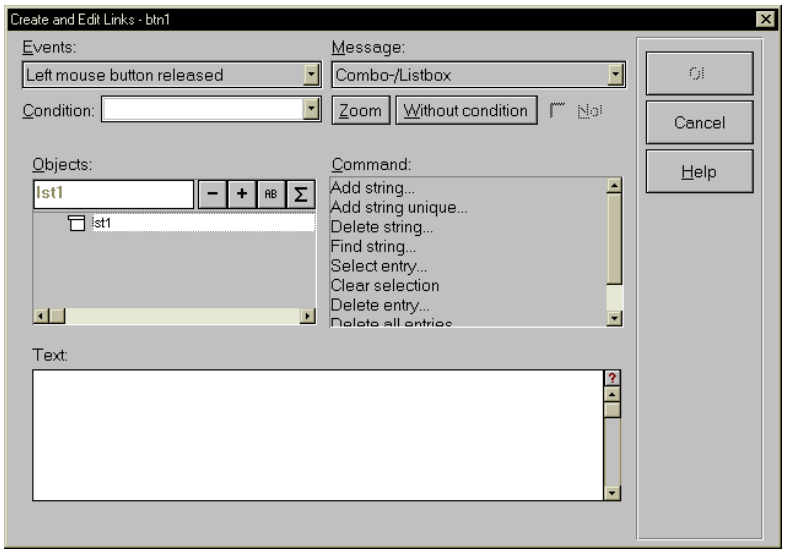
This function lets you completely erase the rows of a data table

This group contains one command, which deletes all rows. Column operations are still possible, such as insertion and deletion, but no rows exist. The display of the data table will only retain the column headings and the previous size.

- The Data table component is described on page 184.
- More information about events specific to this component can be found on page 248.

Combo- / List boxes

This message is listed in the collection of standard messages, but it will only display its options if a combo-/list box has been placed in the application and selected in the left hand side of the dialog box. The available options are:



Depending on the configuration of a combo-/ list box, a predefined selection of commands appears

The field labeled Formula at the bottom of the dialog box is parsed at run time. It will accept either literals enclosed in double quotation marks or standard parser statements.

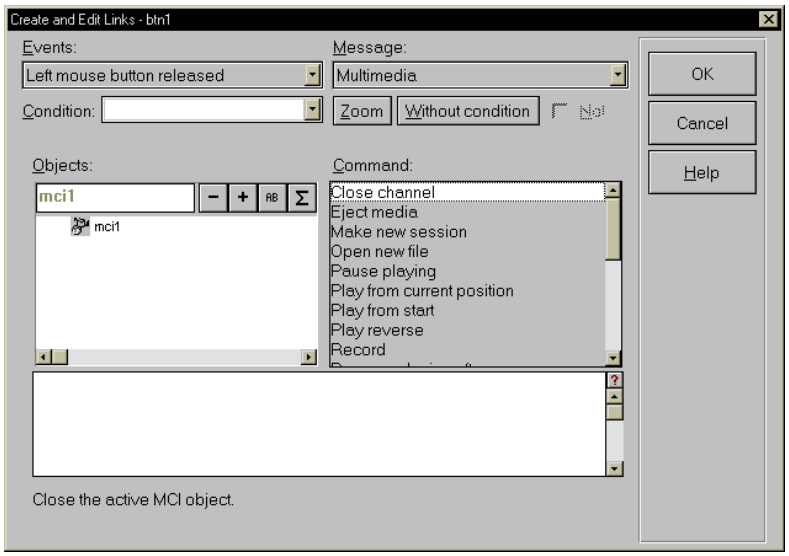
Option	Description
Add string	This option will add the string referenced in the field at the bottom of the dialog box to the combo-/list box selected at the left hand side of the dialog box. If the parser statement evaluates to multiple lines, e.g. by inserting crlf strings or by fetching the contents of a multi line input field, multiple insert operations will be performed.
Add string unique	This option will add the string referenced in the field at the bottom of the dialog box to the combo-/list box selected at the left hand side of the dialog box, only if the string is not already contained in the combo-/list box. If the parser statement evaluates to multiple lines, e.g. by inserting crlf strings or by fetching

	the contents of a multiline input field, multiple insert operations will be performed if the lines are distinct.
Delete string	This option will delete an occurrence of the string which matches the string referenced in bottom of the dialog box. Regardless of the current position in the combo-/list box, the list will be scanned from the top of the list. The comparison is case insensitive.
Find string	This option will find the first occurrence of the string which matches the string referenced in the bottom of the dialog box. Regardless of the current position in the combo-/list box, the list will be scanned from the top of the list. The comparison is case insensitive.
Select entry	This option will select the row in the combo-/list box which is specified in the bottom of the dialog box. The option expects a number denoting a row number. Row numbers are 1-based, meaning that the first row is number 1. Attempting to select a row number 0, will result in no row being selected. Attempting to select a row number greater than the number of existing rows will leave the last selected row still selected.
Clear selection	This option will remove the highlight of any selected row. If multiple selection has been defined for this component and multiple rows have been selected, then this operation will remove all selections.
Delete entry	This option will delete the row in the combo-/list box which is specified in the bottom of the dialog box. The option expects a number denoting a row number. Row numbers are 1-based, meaning that the first row is number 1. Attempting to delete a non-existing row number will cause no row to be deleted.
Delete all entries	This option will delete all entries in the specified component, regardless of any selections.
Directory	This option will fill the specified combo-/list box

	with the contents of the directory referenced in the bottom of the dialog box. The valid syntax for the specification is: "x:*.*" where x: is a valid drive; and *.* are wildcards for file names. You may also specify any other valid file name portion, such as *.BMP or XYZ*.*. The entry must be enclosed in double quotation marks. Directory delimiters must always be two backslashes. The field is parsed so that a component name or a valid parser statement can be substituted for a path specification in the form of a literal. If the component specific attributes of the component were not set to allow the display of files, this message will automatically set the attribute, although it will only display file names (not drives and/or paths). In order for drives and/or paths to be displayed, the corresponding attribute settings must have been made. If you have selected collapsed list box, then this operation will have no effect.

MCI

If you have multimedia capabilities installed on your system, then the MCI component will register a set of its own messages:



Yet another way to control multimedia devices - commands for a MCI component

The MCI component also registers a set of useful parser functions. See page 472.

Command	Description
Open File	Open a media file. This command is applicable only to media that is associated with a file, e.g. Video for Windows or Wave Audio. If the value input field at the bottom of the dialog contains or evaluates to a valid file name, then this file is opened. Otherwise a file open dialog box appears, which allows you to select a file. The system automatically loads a media driver that is suitable for the given file extension.
Close Channel	Closes a previously open media. You should use this command only if you control a device's behavior by means of special media commands. Usually, a device is automatically closed when its page becomes deactivated.
Play	Plays a previously opened media from the current position or from start, if it has not been played before. If the value input field contains a numeric value, playing starts from the given

	position. The units of this value depends on the type of media and its configuration (see the commands Use Frame Mode and Use Time Mode).
Play from Start	Plays a previously opened media from start.
Play Reverse	Play the media backwards, if applicable to the particular media. Note that Video for Windows files usually accomplish reverse playing showing single frames (step mode).
Stop	Stops playing the media. Playing resumes either through the Play or the Resume command.
Pause	Pauses playing of the media. The function is similar to the Stop command. However, if hardware media is directed by the media driver, this command differs from the stop command in that it sets the device in 'stand by' mode. For instance, a video disc player will keep the video heads running, while the Stop command will shut down the player.
Eject Media	Ejects the media, if applicable. Generally, CD audio drives and video disc players can eject the media.
Make New Session	This command creates a new media session. It is applicable only to media that has the ability to record. The value input field contains the name of the media device (e.g. 'waveaudio'). Note that Video for Windows movies generally cannot be recorded through the MCI interface.
Save File	Saves a previously recorded media stream to a disk file. The value input field contains a valid file name. Make sure that the extension of this file is suitable for the current media type. If this field is empty, a file save dialog box appears.
Use Frame Mode	Sets the media to use Frame Mode. (e.g. applicable for Video for Windows to return or set positioning information using video frames).
Use Time Mode	Sets the media to use Time Mode. (e.g. applicable for devices to return or set positioning information using the elapsed time from start).

Resume	Resumes playing of the device after a previous Pause command.
Seek To	Sets the position of the currently open media to the numerical value given in the input field at the bottom of the dialog. The units depend on the particular configuration of the device. (see the commands Use Frame Mode and Use Time Mode).
Send String	Can be used to send command strings to the media device (which must have been opened through either the Open File or the Make New Session command). This message allows full control over the device. Please refer to the documentation of the media specific driver.
Set Volume To	Sets the output volume to the level defined by the numeric value in the input field at the bottom of the dialog. Default volume is 1000 (= 100%). A value of 500 means half volume.
Set Speed To	Sets the playing speed to the level defined by the numeric value in the input field at the bottom of the dialog. Default speed is 1000 (= 100%). A value of 500 means half speed.
Set Repeat Mode	A value of 1 in the value input field indicates that the media has to be played repeatedly. 0 disables repeat mode.
Single Step	This command moves the current position of the media by a certain step relative to the current position. A value of 1 in the value input field means 1 step forward, -3 means three steps backwards.

➤ For more information about the MCI component see page 221.

Output Page

This component is the entity generally used to print information. Therefore, the specific messages it can deal with all relate in some manner to its printing.

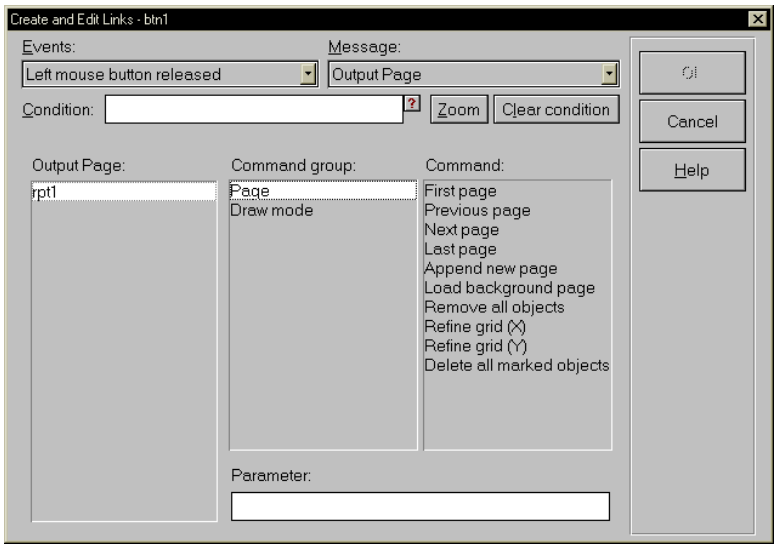
Since an output page is more or less another form of a mindmap page, components can be placed on it in the same manner as they are on mindmap pages.

Since this component is a standard component, the entry in the message list will appear, regardless of the placement of such a component in the application. The messages for the output page component are only available though, if an output page component has been placed in the application.

There are two groups of commands available for an output page component:

- Page
- Draw mode

We will first explore the Page group:



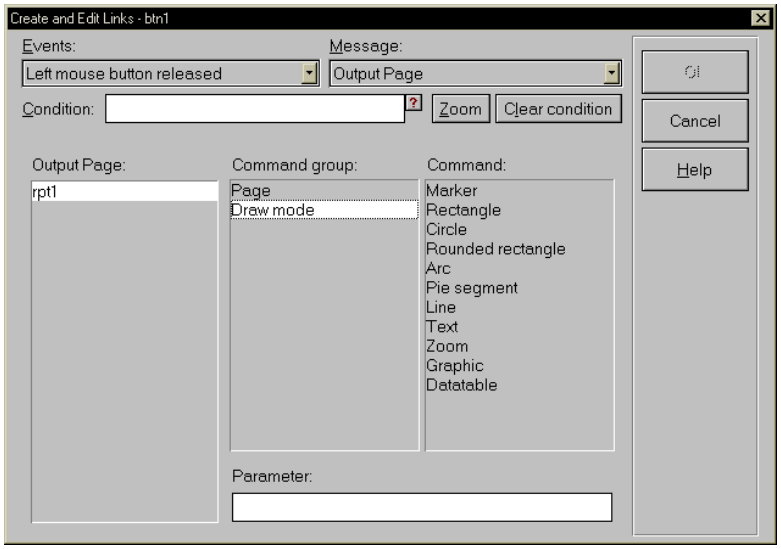
This dialog lets you manipulate the components on an output page

This group contains several commands:

Command	Description
First page	If the output page component has more than one page, this command will position to the first page and display it. If only one page exists, this command will do nothing.

Previous page	If the output page component contains more than one page and it has not been positioned to the first page, then this command will position it to the previous page.
Next page	If the output page component contains more than one page and it has not been positioned to the last page, then this command will position it to the next page.
Last page	This command will position to the last page. If only one page exists, then this command will do nothing.
Append new page	This command will add a new page to the right of the last page and position to it.

Next let's investigate the Draw mode group of commands:



A user may create new components on an output page at run time

Here we have the following commands:

Command	Description
Marker	This command will allow the user to select components on the output page component. It is equivalent to the arrow on the toolbox when mindmap is in edit mode.
Rectangle	This command allows the user to draw a rectangle on the output page component at run time. The cursor remains in this draw mode, until a different function is selected. The standard procedure for accessing a component's attributes are also valid in run mode.
Circle	This command allows the user to draw a circle on the output page component at run time. The standard procedure for accessing a component's attributes are also valid in run mode.
Rounded rectangle	This command allows the user to draw a rounded rectangle on the output page component at run time. The cursor remains in this draw mode, until a different function is selected. The standard procedure for accessing a component's attributes are also valid in run mode.
Arc	This command allows the user to draw an arc on the output page component at run time. The cursor remains in this draw mode, until a different function is selected. The standard procedure for accessing a component's attributes are also valid in run mode.
Pie segment	This command allows the user to draw a pie segment on the output page component at run time. The cursor remains in this draw mode, until a different function is selected. The standard procedure for accessing a component's attributes are also valid in run mode.
Line	This command allows the user to draw a line on the output page component at run time. The cursor remains in this draw mode, until a different function is selected. The standard procedure for accessing a component's attributes are also valid in run mode.
Text	This command allows the user to draw a text component on the output page at run time. The cursor remains in this draw mode, until a

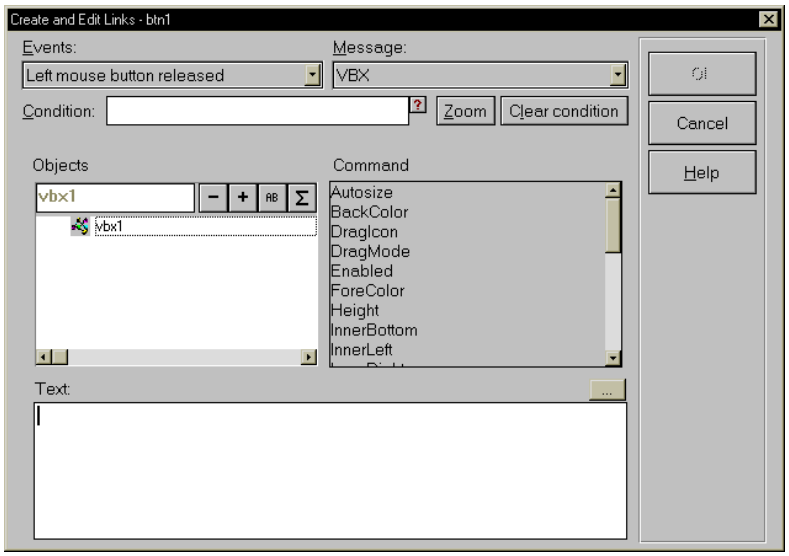
	different function is selected. The standard procedure for accessing a component's attributes are also valid in run mode.
Zoom	This command displays the zoom toolbox, containing zoom in, zoom out and 1:1. The user can perform the zoom operations on the currently displayed output page component at run time.
Graphic	This command allows the user to import a graphic on the output page component at run time. The standard procedure for accessing a component's attributes are also valid in run mode.

VBX

This message is available, if you have included a VBX via the **File | Preferences** menu (see page 56). Only if the VBX has been placed in the application, will the actual messages be available, though.

The specific messages a VBX component is capable of dealing with is a function of the component itself. Each VBX has its own messages, so that you will have to consult the documentation accompanying the VBX.

The layout for the VBX message is consistent, independent of the selected VBX component. What differs are the actual commands, which are integral to the VBX component itself.



Every individual VBX component offers a specific list of commands to manipulate the appearance of the control

The field labeled Text accepts the parameters associated with the selected command. Thus it is also dependent on the VBX itself.

➤ For more information about the VBX component see page 226.

Encapsulation (*Client/Server*)

This icon represents the messages associated with the encapsulation component relating to the server instance. It will only be available, if your application includes a server instance.

Once you select this message, mindmap will locate the server and display the server's default message and its API.

The default message which is available for all server instances is <<<Close the Server>>>.

This message instructs the application (subassembly) containing the server instance, to terminate normally.

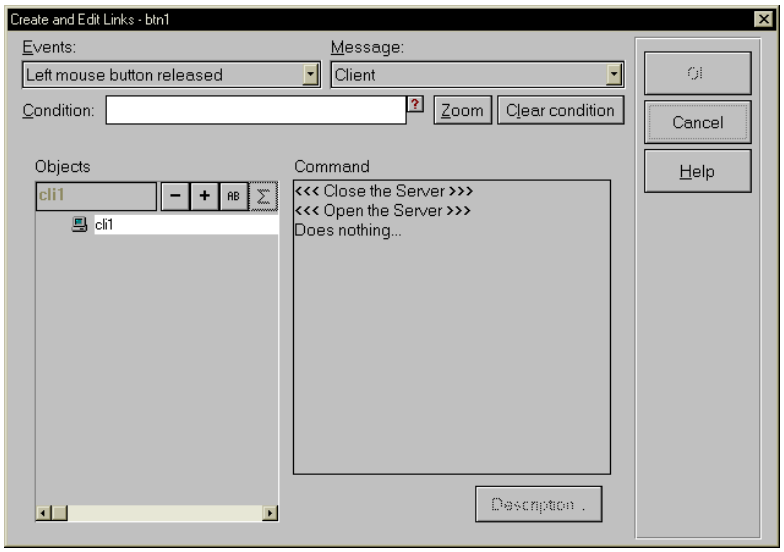
In addition to this default message, the server instance has those messages available that the builder of the instance de-

fined for it. These are the statements (API) that were defined in the process of constructing the server subassembly. Their function is dependent on whatever the designer intends them to do.

This icon represents the messages associated with the encapsulation component relating to the client instance. It will only be available, if your application includes a client instance.

Once you select the message from the list of available messages, mindmap will locate all client instances in your application and display them in the list. When you select one of these client instances, mindmap will locate the physical instance of the corresponding server (encapsulation) instance, open it, and query it for its API. If the associated server instance is not located on the same computer, mindmap will attempt to establish a physical connection to the computer on which the server component resides. It will then logically connect to the component and interrogate its interface. In this case, it might take some time to establish the connection, depending on the connection method (modem, ISDN, network, etc.). mindmap will display a progress bar during the process.

The result of this process is then displayed in the right hand section of the dialog box.



The link dialog for a client/ server component includes predefined commands as well as developer defined ones

The <<<Close the Server>>> message sends this command to the server instance, which causes it to close down. This message is automatically generated whenever the client component, which is connected to the server instance, is terminated normally. It is still suggested that you generate this message in a controlled manner from within your client application.

The <<<Open the Server>>> message must be sent to the server instance, before it will accept any other messages. Without receiving this message, the server instance will remain dormant.

Following these two default messages, are whatever other messages the constructor of the server application intended to expose to the outside world.

Database

This message entry is visible even if a database component has not been placed in the current application. The actual message options are only available if a database has been placed.

In the case of the ODBC driver, the set of commands are identical, independent of the actual underlying database. In those cases in which the database is not a SQL compliant database, the ODBC driver emulates the actions of a true SQL database.

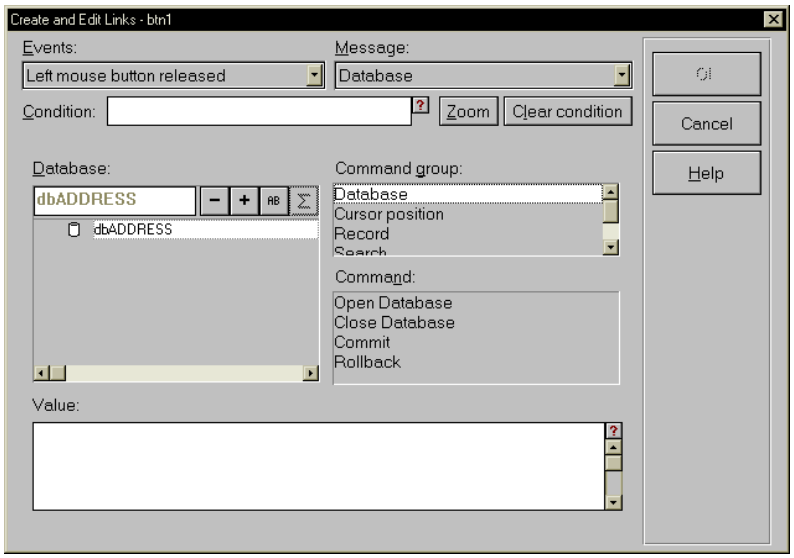
The database message contains five groups of commands. These are:

- Database
- Cursor position
- Record
- Search
- SQL Exec

Each one of these groups has a set of commands. The majority of the commands necessary to interact with a database have been summarized into high-level commands, making the input of SQL superfluous. These commands can usually be augmented by the inclusion of parameters specific to the command. In addition to these high-level commands, mindmap also offers the opportunity of explicitly entering SQL commands, thereby permitting complete control of the database.

Please note that it is beyond the scope of this manual to describe SQL (Structured Query Language) in detail. Despite a standardized group of commands and syntax, many database vendors have created specific extensions for their own databases. Again, it is beyond the scope of this manual to cover all of these variations. Please refer to vendor-specific manuals for more details.

We will begin by describing the first group:

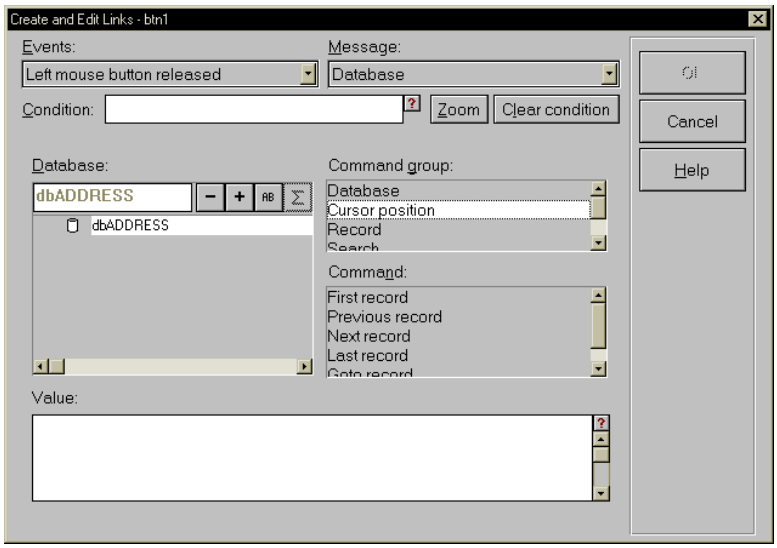


A database offers various commands for changing, searching, or navigating through a database

Command	Description
Open Database	This command will open the selected database. If the driver accepts any additional options, these can be entered in the field at the bottom of the dialog box labeled Value.
Close Database	This command will close the selected database.
Commit	The default setting for a database is auto commit. If this attribute has been deselected, then the process of committing a database operation can be controlled manually. This command will have the database commit the operations. Please note that the particular ODBC database driver must support commit/rollback for this function to be available.
Rollback	The default setting for a database is auto commit. If this attribute has been deselected, then the process of rolling back a database operation can be controlled manually. This command will have the database rollback all

	non-committed operations.

The second group of commands deals with the positioning of a cursor in the database. By definition, the database cursor can be moved to any record in the selected table. Operations such as deletion of records, insertion of records, etc., are then performed at the cursor position.



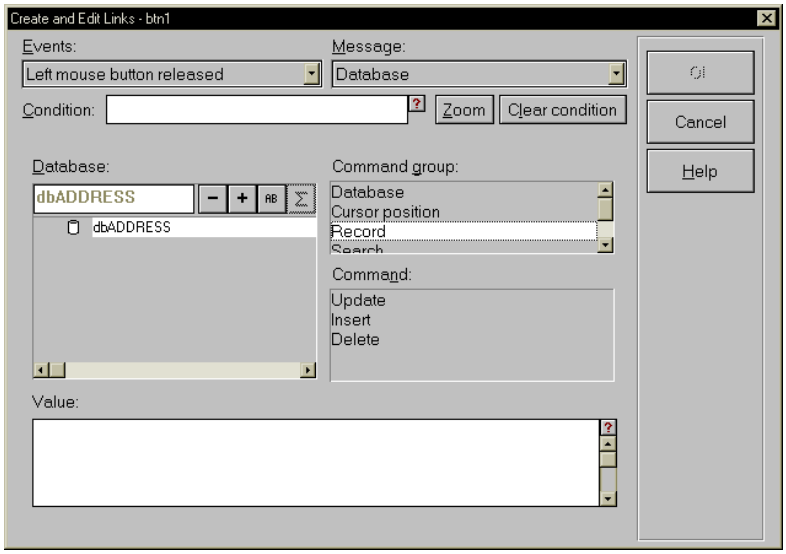
These commands let you browse through a database table

This group contains five commands. These are:

Command	Description
First record	This command will position the cursor to the first record in the table. Record numbers are 1-based, so that the first record is number 1.
Previous record	This command will decrement the record counter by 1 and position the cursor at that point, if it is larger than 1. Otherwise, the cursor will remain at the first record.
Next record	This command will increment the record number by one and position the cursor at that

	point, unless the cursor is at the last record in the table, in which case nothing will happen.
Last record	This command will position the cursor to the last record in the table and set the record number to the last record.
Go To record	This operation will position the cursor to the record specified in the field labeled Value. The Value field will accept either an integer, or a component name which either contains an integer or another component name. The field is parsed at runtime. Database record numbers are 1-based.

The third group of commands deals with write operations to the database. These are:



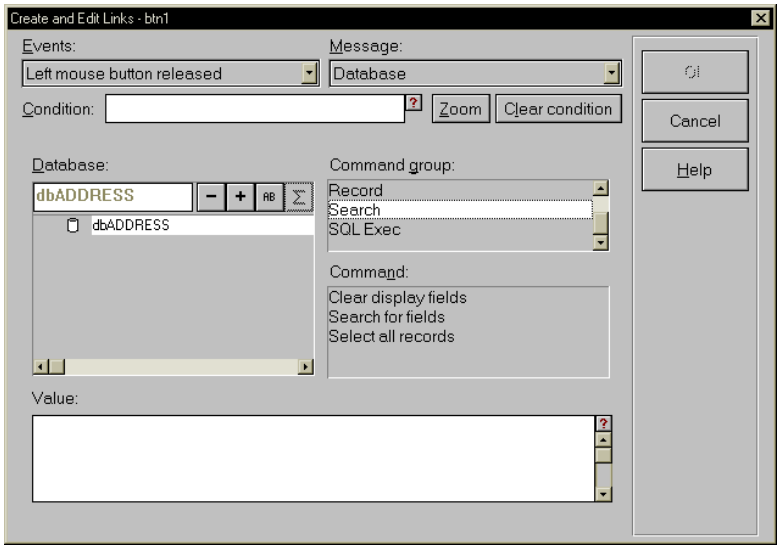
These commands are used to modify records in a database table

This group contains three commands:

Command	Description
Update	This command updates an existing record and writes the contents of the components to the corresponding fields in the database table. The record to which the components are written is determined by the cursor position. The record at which the cursor is pointing at the time the operation is committed, is the record to which the data is written. Which components are written into their corresponding database fields depends on the particular assignment settings. See page 147 for more information. No additional parameter (Value) is necessary.
Insert	This operation writes the contents of the components to the corresponding fields in the database. In doing so, this operation creates a new record in the database. Since relational databases are not kept in any specific order, the insertion occurs at the position of the cursor at the time the operation is committed. Which components are written into their corresponding database fields depends on the particular assignment settings. See page 147 for more information. No additional parameter (Value) is necessary.
Delete	This operation deletes the record the cursor is pointing at, when the operation is committed. Please be aware of the fact that deleting a row from a table may result in serious implications for the database. Consider two tables containing customer information and employee information. An employee who is responsible for a particular customer may be identified in the customer table only by his ID. If you delete the employee, you will leave the corresponding ID in customers without reference to the employee table. This is a violation of referential integrity. If not defined by the underlying database system, mindmap does not take special care

	about the referential integrity of the database and therefore will delete the selected record.

The fourth group of commands deals with queries to the database. These are:



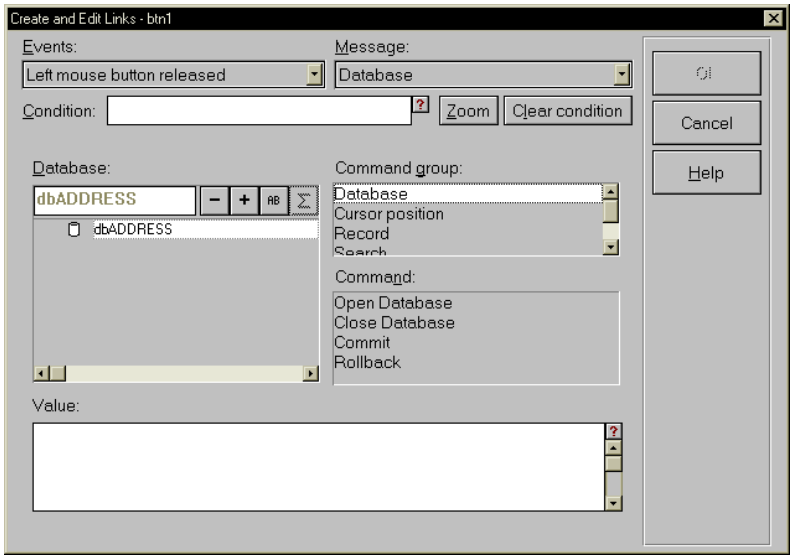
Use these commands to query the database

After search conditions have been entered in other mindmap components, use these commands to query the associated database table:

Command	Description
Clear display fields	All input fields on any mindmap page associated with the selected database will be cleared. This means that their contents will be set to whatever has been previously specified in the corresponding default settings for each input field (Preset). Commonly, a null string has been set, so that the input fields will be blanked. See page 193 for more information about input fields.
Search for fields	The default setting for an input field associated with a database is that it will be included in the

	query. Executing this command will construct a SQL SELECT statement which includes all fields that are set to be query fields. To exclude fields from participating in the query (being included in the SELECT statement), change the assignments in the connection between the input fields and the database table. See page 184 for more information. The exact structure for the SELECT statement is also determined by the search attribute of the corresponding input fields.
Select all records	This command will generate a SELECT * FROM table, resulting in a query which selects all records in the database. If dealing with a large database, be careful in executing this command, due to the possibility of an extremely large number of records being returned.

The last group of commands allows you to expand the existing high-level database commands by means of explicitly generating SQL statements. For those users who wish to generate their own SQL statements, this is the location to do so. The commands included in this group are:



Enter your own SQL statements to gain full control over a database

Command	Description
SQL SELECT	This command expects the WHERE, GROUP BY and/or HAVING clause of a SQL statement in the value editor at the bottom of this dialog. mindmap will automatically prefix the given SQL part with the clause `SELECT <connected fields> FROM<table name>` Typically the given clause could be 'WHERE name like "Smi%".
SQL COMMAND	This message lets you enter and execute any type of SQL statement supported by the database. Use this command to typically issue commands which do not return a result set.
SQL COMMAND with refresh	This message lets you enter and execute any type of SQL statement supported by the database. You should use this command to execute SQL commands which return result sets. Always make sure that the returned result set matches the assignment of mindmap components as defined by the database component. Retrieving data from the result set to undefined or invalid components may give unpredictable results.

Input/Output

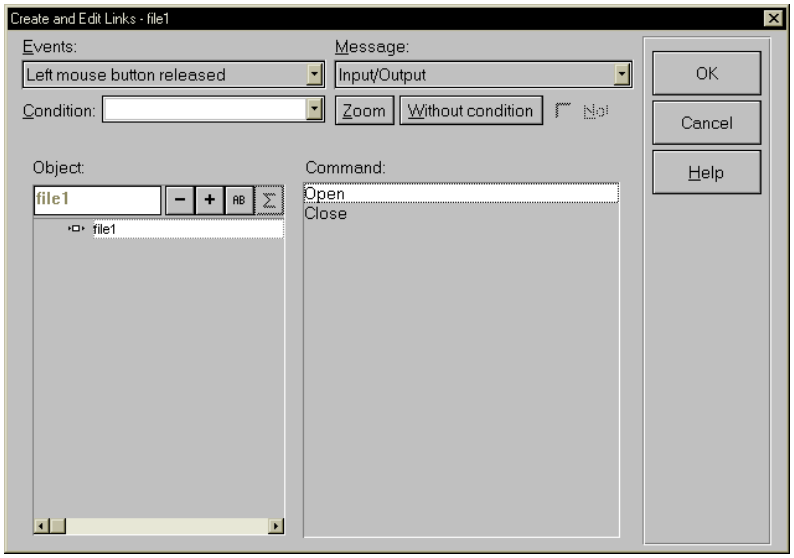
There are two different entities to/from which this command can be applied.

These are:

- File System
- Printer

Interaction with the file system implies that you either wish to read from or write to a file. Therefore, the appropriate messages are:

- Open
- Close



The input/ output component offers two messages to open or close a printer or file component

Note that printing in a network environment the input/output component will generate a new print job, whenever a printer is opened and closed. It is more common to put multiple print operations into a single job. Use these two commands to open and close the print job accordingly.

Select the input/output component from the list on the left side of the dialog box. Then select the message, depending on whether you are reading or writing a file. If you wish to read from a file, then drag&drop the contents of this component to a component which can deal with the type of data you are reading (i.e., input field for alphanumeric data, a bitmap for incoming or outgoing graphic data, etc.). Refer to the attributes of the component to set such things as file name, prompt user for file name, etc.

These two commands cause the file or printer being opened and closed through a link message. Usually, a drag&drop operation, that acts on either the printer or a file, will automatically open and then close the appropriate channel. Issuing multiple drag&drop commands after another, will cause unwanted open and close actions on the printer or file. Therefore, it is more convenient to open the appropriate channel first and close it when no longer needed.

Chapter 7 Using the Client/Server Component

General Overview

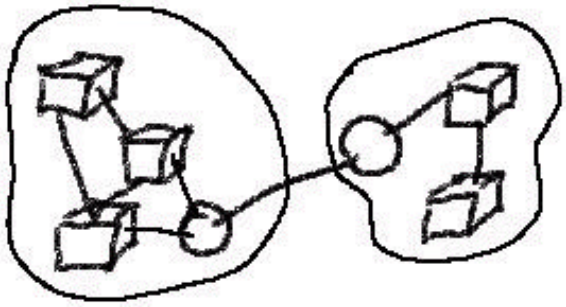
mindmap consists of a set of basic building blocks -- the components. These components have attributes which the developers can set to certain conditions. The fundamental building blocks are then linked to one another via events and messages.

The client/server component expands this metaphor to now include higher level building blocks -- subassemblies. A developer can use the basic components to assemble a functional entity. This entity can then be used in other situations as though it were a basic building block.

In chemistry, this would be comparable to arranging elements to form molecules. In the electronic industry this is analogous to combining chips to form an adapter board which plugs into the motherboard.

It is important to note that this approach allows the inner working of the subassembly to be masked from the outer world. The subassembly presents itself to the rest of the world via a well-defined interface. Molecules have docking stations at which other elements or molecules can attach themselves. The interface between the motherboard and adapters is the bus (such as ISA, EISA, PCI, etc.).

In the mindmap environment, a special component -- the client/server component -- functions as the sole interface between the subassembly and the other components of an application. Thus, a c/s component encapsulates the complexities of the subassembly.



Basic building blocks are combined to subassemblies

This construct leads to a number of advantages:

- Teams can develop applications with minimum interaction necessary between the groups.
- A large application can be broken up into more manageable units (subroutines) and these subassemblies can be reused from within other applications.
- Applications can initially be built locally and then deployed remotely, or they can actually be built at remote sites.

In the case of team development, it is possible to have various individuals, or groups of individuals, work more or less independently of one another on the completion of an application. In this case, each individual would be assigned a specific task which would be realized in the form of such a subassembly. Each subassembly would make itself accessible via its encapsulator (c/s) component. Members would only have to agree on the functionality required from a subassembly and how it would be accessed.

When a large application is to be developed, it often becomes necessary to decompose the complexity by defining blocks of functionality. In traditional programming environments, this was done by means of subroutines and procedures. In the case of mindmap, the functional entities are assembled into a collection of basic building blocks and subsequently wrapped up in a subassembly. The c/s component (encapsulator) becomes the

point of interaction between the subassembly and the rest of the world. If done properly, meaning with sufficient abstraction, this subassembly can be used in other applications to perform the same task. This is what was referred to as code reusability in the classical programming environments. In the mindmap context, it is called functional reusability.

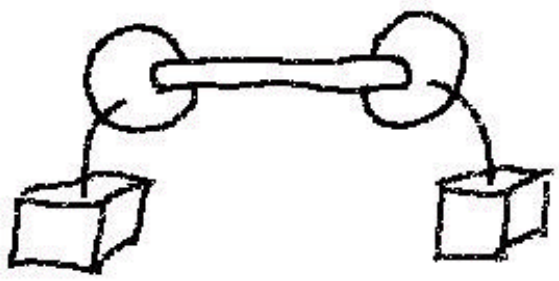
In conjunction with the Internet or company-wide intranets, applications or portions thereof can be posted on easily accessible servers and downloaded at the time of execution. The machine which is to execute the application has the mindmap environment installed and only needs to download the application part, which in general is very small and can thus be easily transported. This makes for easy maintenance of central applications.

If you are thinking in terms of disparate computers, then mindmap's client/server technology can be used to build highly distributed applications. A subassembly can be built so that it will run on a remote computer. If and when the functionality embedded in the subassembly is needed, a connection is established and the functionality is accessed. The construction of such distributed types of applications can be undertaken locally on one and the same machine and then deployed when complete, or can actually be done on the dislocated machines and merely connected for testing and deployment purposes.

When two subassemblies are connected, the channel running between the two entities can transport two different types of information:

- links; that is, events and/or messages,
- data.

Links are passed just as they would be between two components in any other mindmap application. Data is passed in a seemingly somewhat more limited manner.



How two subassemblies communicate with each other

If component A wishes to send a message to component B, it actually has to send it to encapsulator component A. This component in turn receives an event, to which it then creates a message. This message is then sent to encapsulator component B. This encapsulator receives the event, generates a message and sends it to the component B. Passing data between two components residing in different subassemblies functions by the same token. Instead of sending data directly to the other component, it is forwarded to the intermediary agent -- the encapsulator component.

In both cases -- exchanging events and messages and passing data -- the actual names of the events/messages and data do not have to correspond to one another. The developer of one subassembly -- by definition -- does not have any control over any of the aspects of the other subassembly. This in fact would actually defeat the purpose of encapsulation, where the developer of a subassembly had to rely on his/her knowledge of the inner workings of any other subassembly.

At first, this might seem overly complex, but with a little practice it will become as easy to use as is the simpler relationship between conventional components. It is by communicating with an agent, instead of directly with the receiving component, that is the basis for the distribution of application functionality and for the encapsulation process.

Special Conditions

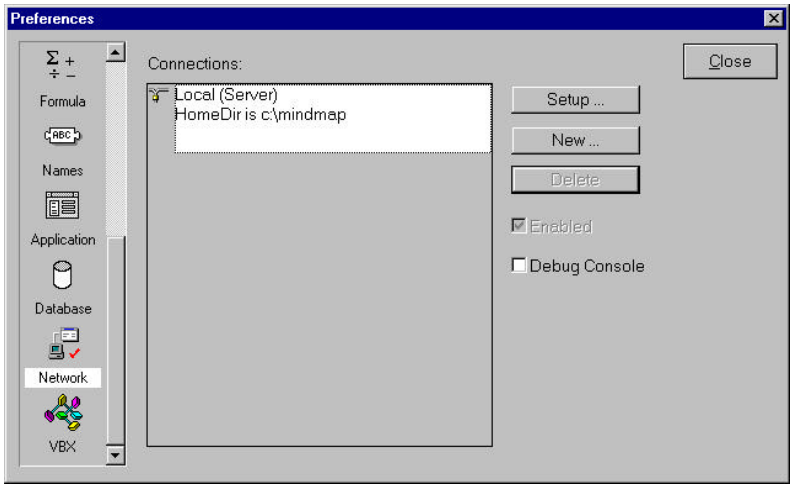
Local Server

The process of building applications based on subassemblies is always the same, independent of where the various subassemblies reside -- locally or remotely. In contrast to remote subassemblies, the facility to access local ones is installed by default. This is due to the fact that no specific communication channels have to be established. The computers on which remote subassemblies (server) reside must be installed in mindmap. This process is described in detail in the next segment.

This default setting allows the immediate construction of applications based on subassemblies.

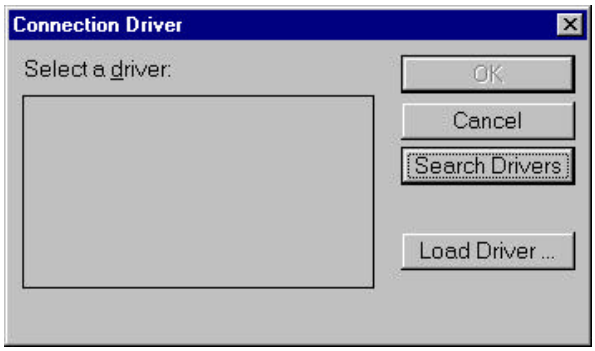
Installing Remote Servers

If you wish to access facilities residing on remote systems, you must first register the host system it resides on. To do this, select the menu option **File | Preferences**. Scroll down to the option Network (see page 55).



A connection to local server subassemblies always exists

Next click on the New button and continue to install a new host.



Search or load various communication drivers

It is easiest to have mindmap search for its communication drivers automatically. To initiate this process, click on the Search Drivers button. mindmap will search its working directory and will return all drivers it is able to locate. If you wish to manually locate the drivers, click the Load Driver… button. You will be displayed with a file selection dialog box. Navigate to the directory containing the appropriate driver and select it.

Both drivers require the underlying protocol stacks to be properly installed and that they are functioning. Since this is under control of Windows, please consult your Windows user manual for proper configuration. You should not attempt to install a driver if the underlying network support is not available.

mindmap ships with two different communication drivers:

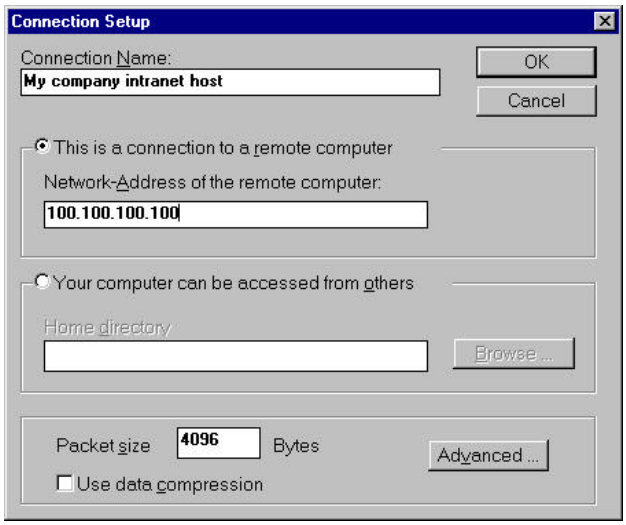
- a TCP/IP connection (MMTCP.DLL),
- a Windows for Workgroups connection (MMWFW.DLL).

The first driver allows two systems to communicate via TCP/IP. This protocol is common on many LANs, as well as on the Internet.

The second driver can be used on a LAN which uses Microsoft's Windows for Workgroups protocol.

Once mindmap has loaded the drivers, select the appropriate driver by clicking on it.

In the case of the TCP/IP driver, you will see the following screen:



This dialog lets you configure a network connection using TCP/ IP

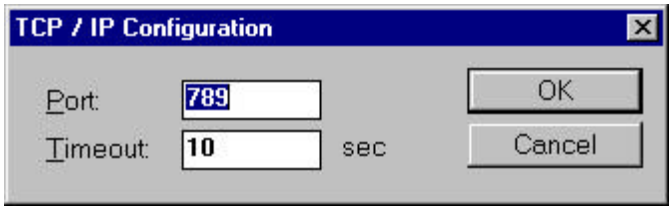
At the top, you must enter a name by which you can address the host computer. This name is only used internally, so feel free to enter any name, as long as it fits into the field.

The second entry is more critical. It is the actual TCP/IP address by which your computer can access this host. You can either enter an actual TCP/IP address or its name. The latter is only a valid entry, when the host you are attempting to connect to has been registered with a name server (DNS entry) or through a proper configuration in the file HOSTS. In this case you can enter the name under which the server has been registered, such as WWW.MYHOST.COM.

The third section of this dialog box will be dealt with, when we describe how your own computer can define itself as a mindmap server.

The fourth entry concerns the size of the packets to be transferred. By default this is set to 4096, since it is assumed that compression will be used. It is also the maximum packet size if compression is employed. If you choose to transmit data without compression, then the maximum setting is 32KB.

In some instances it might be necessary to be able to set additional parameters. These can be accessed by clicking on the Advanced button.



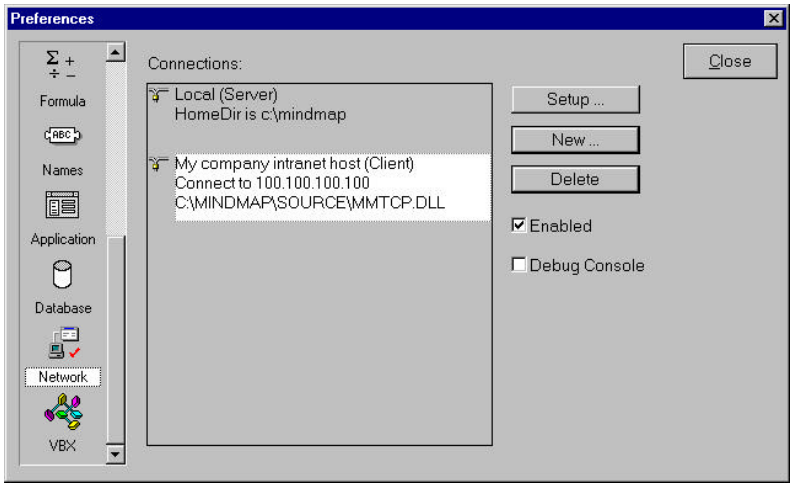
Advanced settings for TCP/ IP

The port settings on the client and on the server side must be identical, otherwise a transmission will not occur, since the data 'bypasses' the transmitter and receiver respectively.

The first entry allows you to specify a port address. The TCP/IP protocol uniquely identifies a remote host by its address (which identifies a specific computer) and a port address (which identifies a specific program running on this computer).

The second permits the entry of a time-out after which mindmap will disconnect, if the remote computer does not answer. If you are using a remote connection over a relatively slow transfer media (such as a modem), you will have to increase this value.

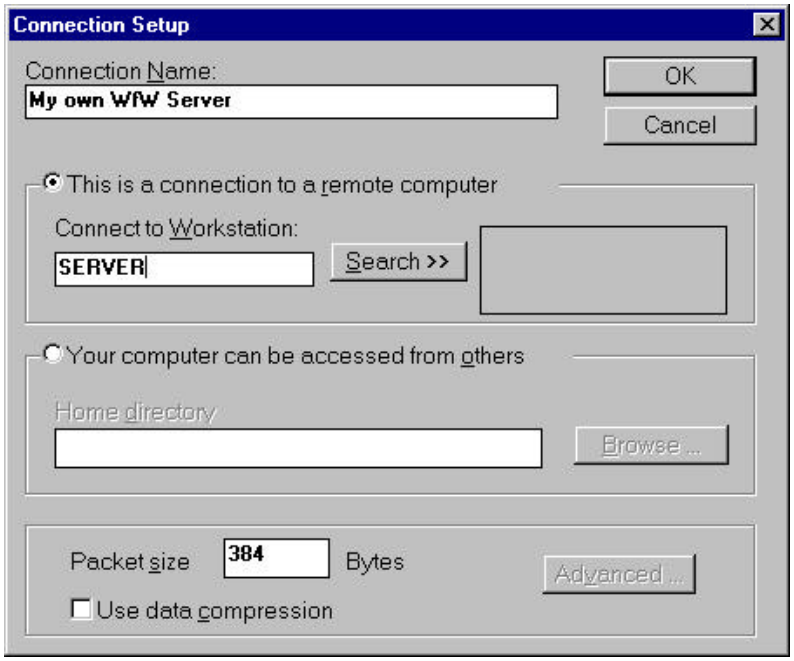
Once you have completed the entry of the required data, mindmap will register the server.



After all options have been set, a new TCP/ IP connection exists

These have been the steps necessary to register a TCP/IP based remote server. In order to register a Windows for Workgroups server, more or less the same steps are required.

Begin by selecting the New button. Then click on the Windows for Workgroups option. You will be presented with the following screen:



Configuration settings for a Windows for Workgroups connection

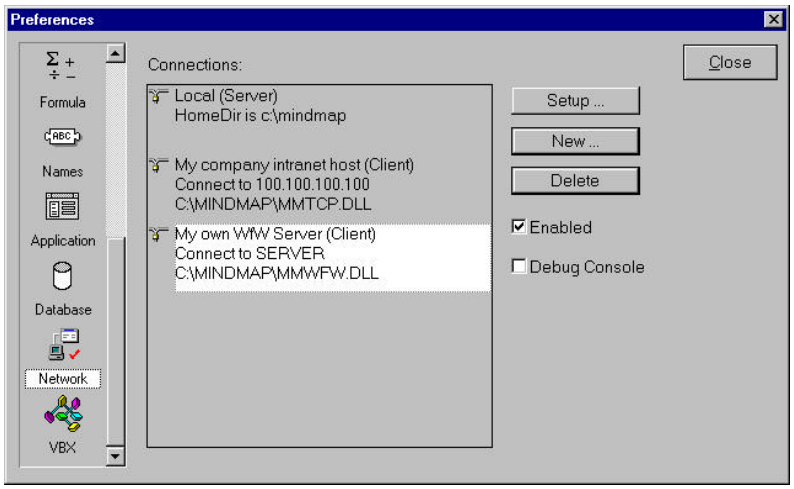
mindmap uses a Windows internal communication facility called Mailslot. Usually the Mailslot API uses the NetBIOS or NetBEUI protocol to transfer information. Make sure that this protocol is installed and properly configured. Due to a restriction in the Mailslot API, no packets larger than 384 bytes can be transferred.

The first entry again is intended to enable you to identify the host computer. Enter any character string you feel allows you to easily recognize the host. This is the name by which the host computer will be listed in the selection list.

Next, you must specify the actual workstation name as it has been defined in Windows for Workgroups. If you are not sure, press the Search button and mindmap will query the appropriate files and return a list of recognized workstations on the LAN. Assuming that you recognize the targeted workstation, select it and the name will be transferred to the field.

The packet size has its default setting of 384. Data compression is also selected by default.

Now the Windows for Workgroups server has been registered and will appear in the list of available hosts.



List of network servers registered through the properties dialog box

The data for all registered hosts is stored in a file called MMNET.INI. This file is located in the mindmap working directory. We suggest that you do not manually edit any of the entries in this file.

You can register any number of servers, which will all appear in the list. As will be described in subsequent sections, mindmap can establish connections to any number of server over one and

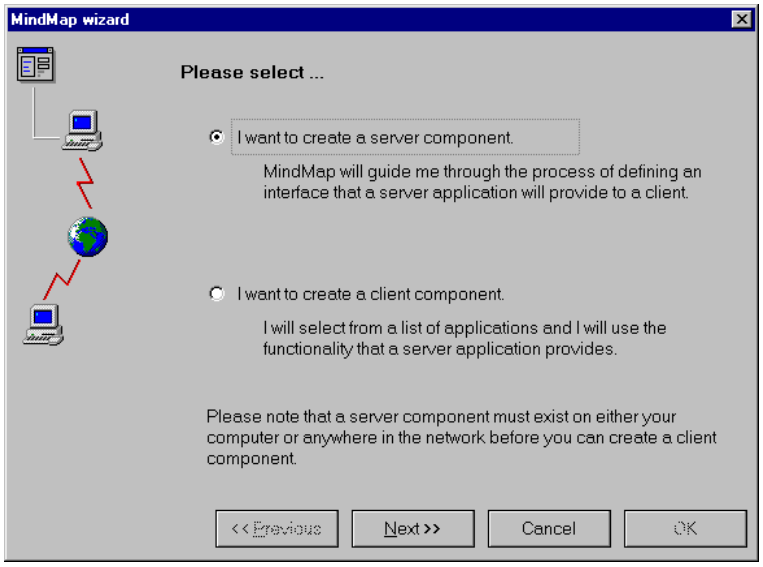
the same protocol. This means that multiple servers will share the same logical and physical connection. The TCP/IP and WfW connections are able to multiplex logical connections on top of physical connections.

Creating a Client/Server Application

Defining a Server Subassembly

The first step is to build a server instance of a subassembly. This server will be called by a client instance.

Begin by opening a new file. Select the icon representing the client/server component. Even before you can place it on the screen, a wizard takes control. This wizard will initially display two informational dialog boxes. Read through these, acknowledging each by clicking on the Next button. You will then be offered the following dialog box:



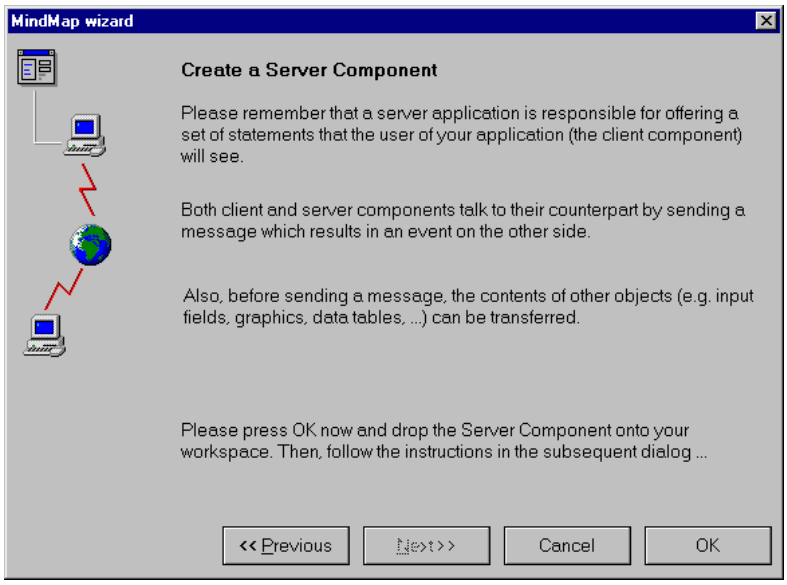
Specify which component you want to create

As already mentioned, since this is the first client/server type application you are presumably assembling, you must select

The terms 'Client' and 'Server' are actually misnomers. mindmap supports the concept of peer networks. Any subassembly can communicate with any other subassembly, given that both have an encapsulator component placed. One subassembly initially calls the other subassembly and is therefore referred to as the client, whereas the called subassembly is designated the server subassembly.

the first radio button. This allows the creation of the server part. Continue by clicking on the Next button.

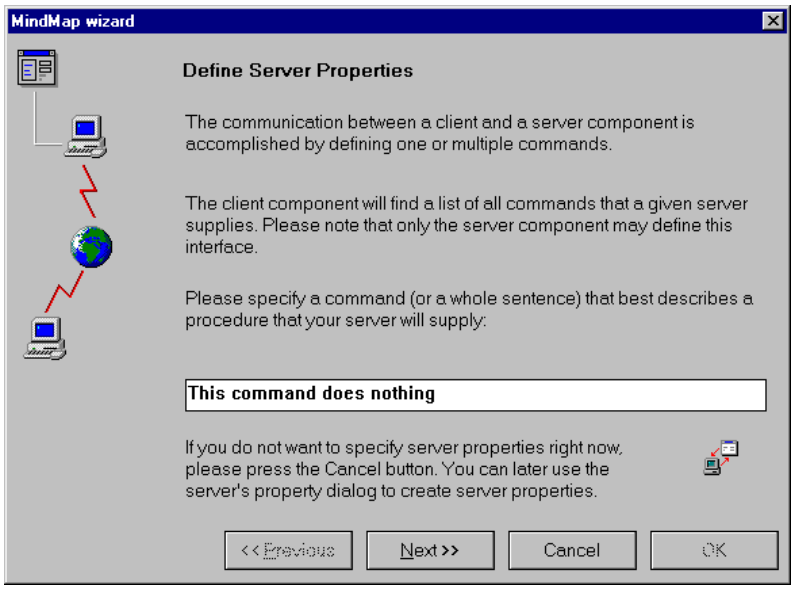
This will take you to the next screen:



Some information on how to continue

This screen contains some additional information, just to remind you of the process of assembling client/server applications. Accept this wizard screen by clicking on the OK button. The cursor will change its appearance to reflect that an encapsulator component is to be placed. Move the mouse to the screen position at which you would like to place the component and click once on the left mouse button. mindmap will now place an indented square on the screen, representing the server instance.

Immediately thereafter, the wizard will take control again and present you with the following screen:



This command will be visible to the corresponding client component

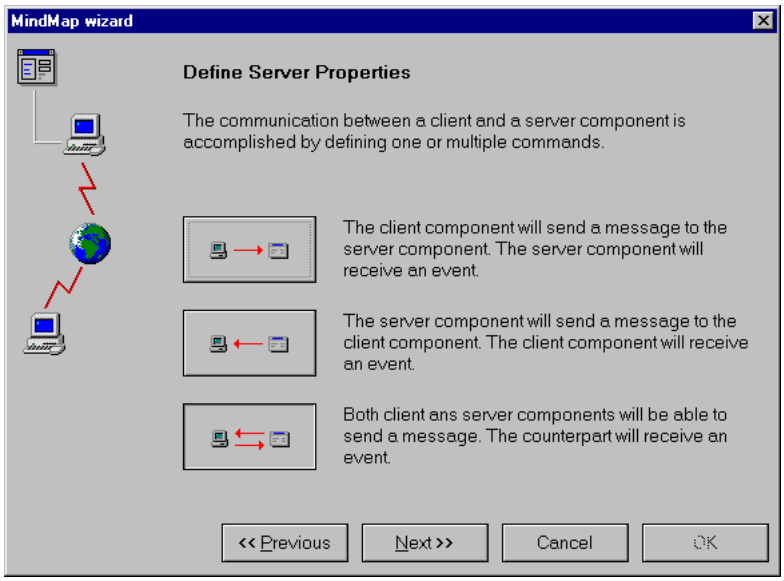
The commands have been entered here with quotation marks. This has been done to make them easily distinguishable from the other text. Within mindmap, do not enter such characters - simply enter the command - without delineators.

This statement will be viewed by the calling client. It should express the actual function associated with it. In this case, no function is attached to it. In a real application, something like

- 'Initiate database query', or
- 'Retrieve picture'

would be entered into this field.

After the first command has been entered, the next screen will appear.

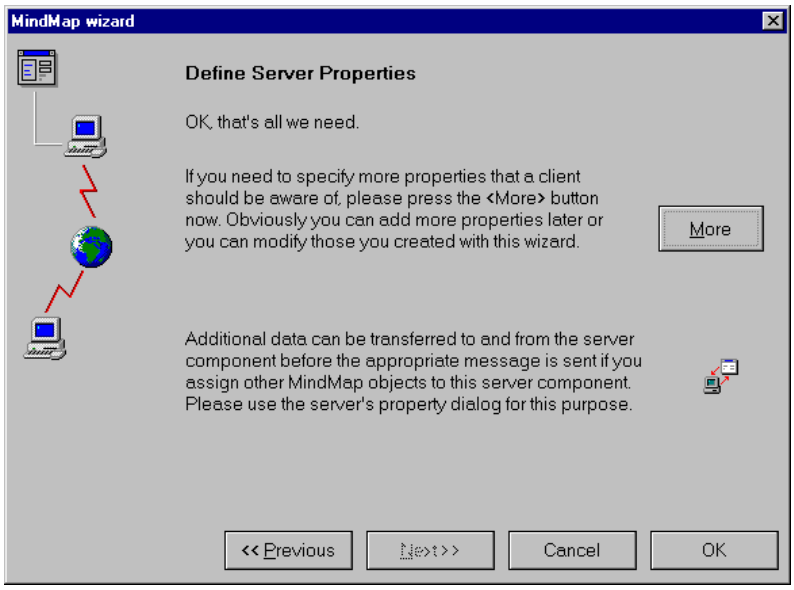


You can select the direction of the information flow

In most real applications, the flow of the 'flow' is critical, so to say. It is strongly advised to give considerable forethought relating to the 'direction of the connection'.

This screen allows a selection regarding the 'direction of the connection'. What this determines is who will be generating a message and who will be receiving an event. The default setting is the last button, reflecting a bi-directional flow.

For the time being, retain the default setting and click on the Next button. This results in the following screen.



Either create more commands right now or begin to work with the new component

Since we will not be entering additional commands at this time, please click on the OK button to exit the wizard.

Control will be returned to the mindmap screen and you can again see the indented box. It should contain the text Server.

If you haven't already done so, please place it in the top left corner of the screen. Now press the screen size button of this little 'application'. Don't size the general mindmap window - only the window associated with the little application. Now size the application window so that is approximately the size of the top left quarter of your visible screen area.

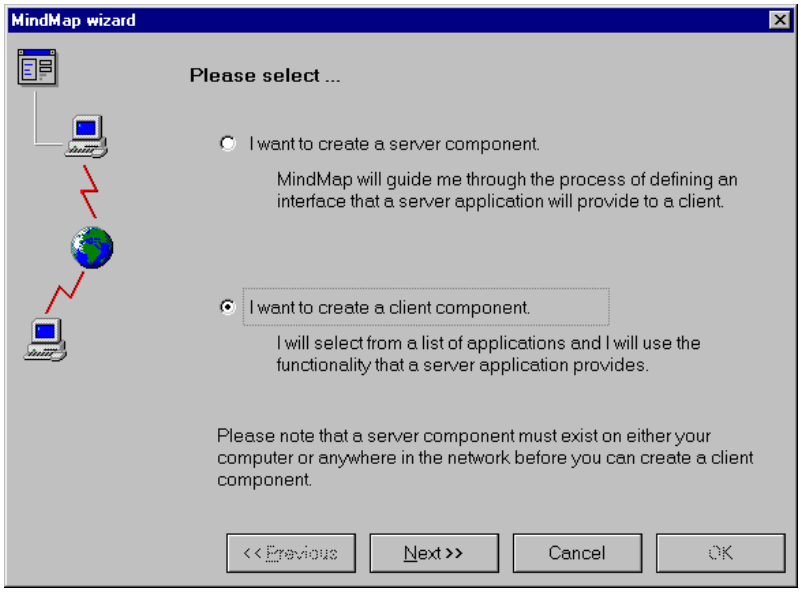
Now save it into the working directory of mindmap, calling it SRV1.MM.

Once this has been accomplished, close this application, but leaving mindmap still active. Open a new file, which we will turn into the client application.

Defining a Client Subassembly

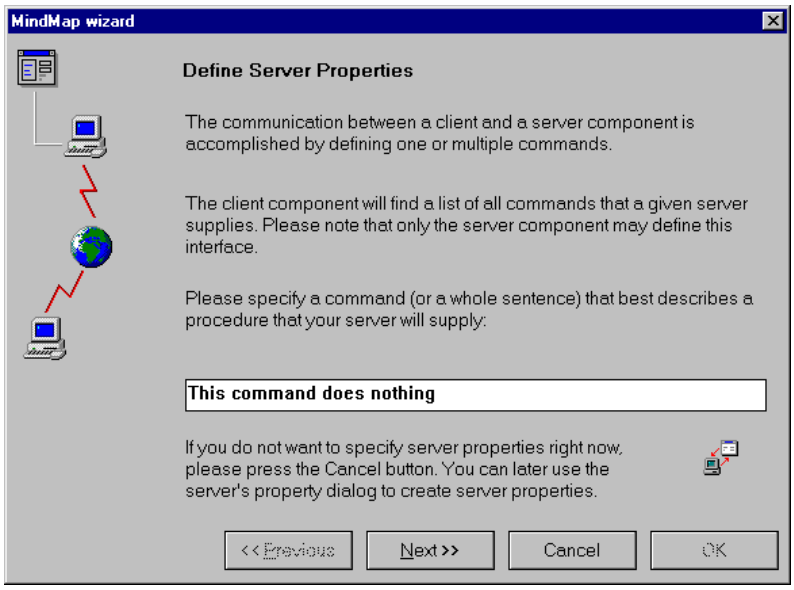
This client application will do nothing else but call the server application SRV1.MM and subsequently close it again.

Begin by again selecting the encapsulator icon on the toolbox. As expected, the wizard will take control and present itself with the now familiar informational screens. You will be offered the following dialog box. Select the radio button associated with building a client component.



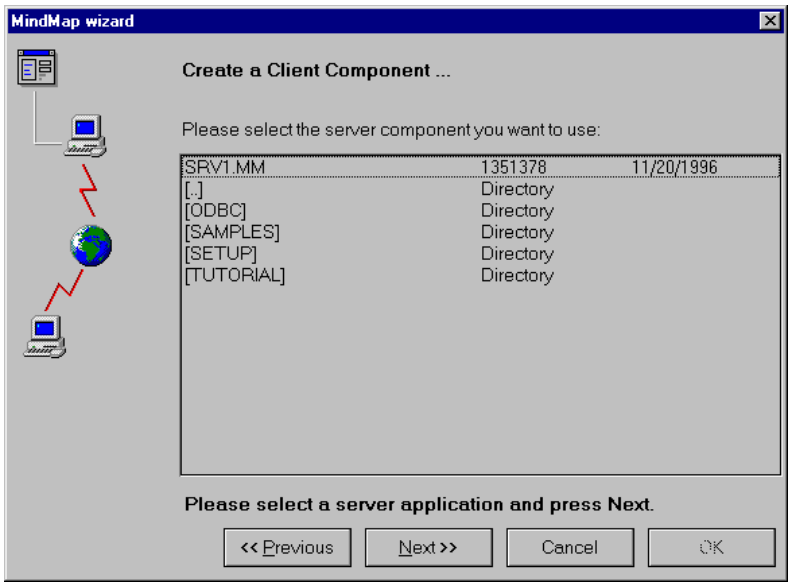
Again create a new component, this time select the client component option

Leave this screen by clicking on the Next button. The wizard will offer the following screen:



Select the location of the corresponding server subassembly

The wizard automatically defaults to the local computer setting. Since we do have other servers registered, they appear in the list on the lower part of the dialog box. We will not be connecting to them, so retain the default setting and click on the Next button. This takes you to the next wizard screen.



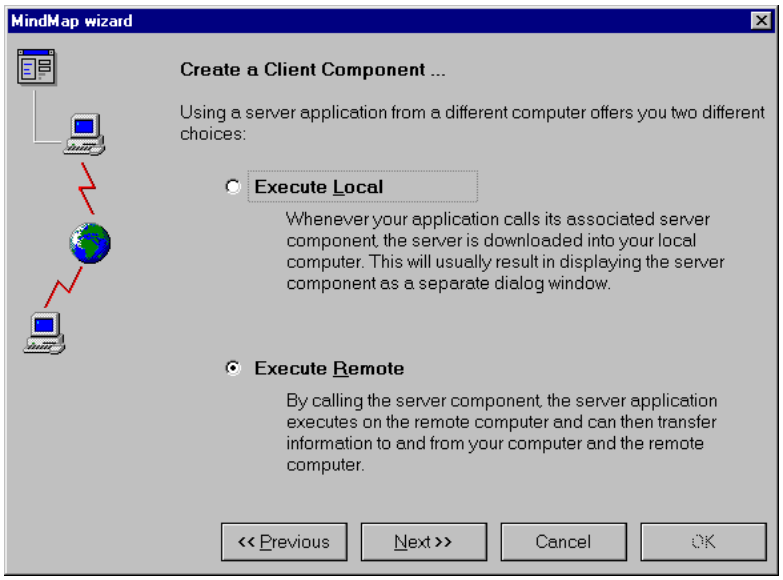
Once connected to the server, you will find the available server subassemblies

Since mindmap does not distinguish applications being as server applications or not, all applications are listed in this window. We strongly recommend that you keep all your server application in a common directory and make it the home directory.
You can change the home directory by accessing the File | Preferences | Network option (see page 55).

mindmap will look for all potential server applications - meaning .MM files -- residing in the mindmap working directory. These are displayed in the list. In addition, directories within the current working directory are listed and can be navigated to by double-clicking on them in the list.

Select the server application you just completed, highlighting it in the list. Finish this screen by clicking on the Next button again, or by double-clicking on the file name.

If you have specified in one of the previous screens that you wanted to access an application from a remote computer, rather than your Local Server, you will have been asked whether you want the server application to be executed on the remote computer or have it transferred to your computer for execution.

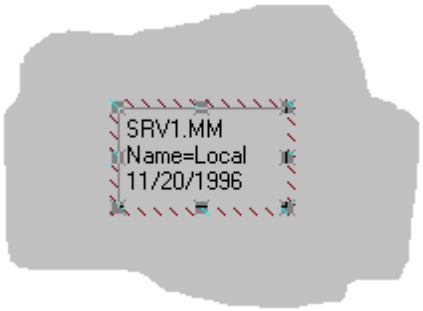


If connected to a remote computer, you must specify where the server subassembly is supposed to execute

It is obvious that a server subassembly has to be especially designed in order to be able to run remotely. A subassembly that requires user interaction is certainly not a good example for a remote server.

You will view an informational screen which you should read and then acknowledge by clicking on the OK button. The control will return to the mindmap desktop and the appearance of the cursor will have changed to signal that you can now place the client instance of the encapsulator component.

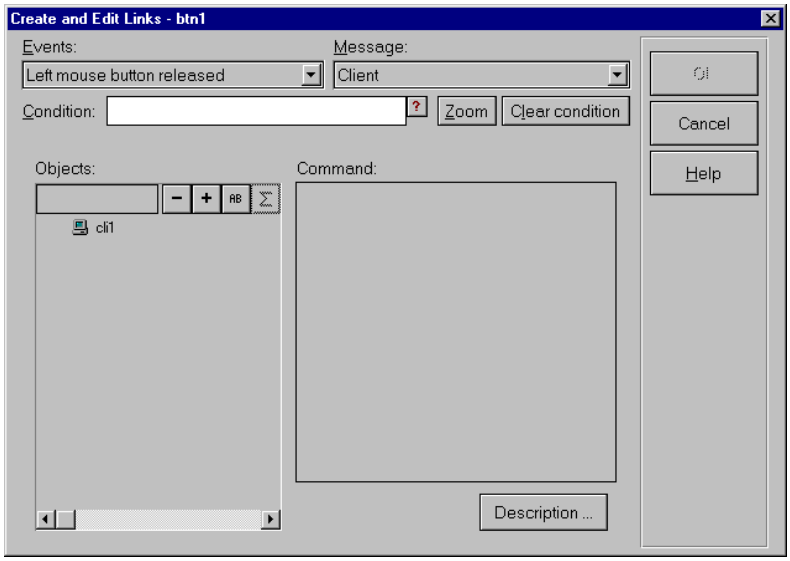
Click once, anywhere on the screen, and the indented rectangle representing the component will be placed. Once that has been done, you might wish to select the component and increase its size somewhat to view some of the information displayed in it.



In principle, we have now established a connection between two mindmap applications. Let's save this client application and then proceed to make it actually interact with the server application. Save it under a file name such as CLI1.MM, preferably into the mindmap working directory.

Assuming that the CLI1.MM file is still open, let's create a command button to open the server application and one command button to close it again.

Place a command button on the desktop and call the linking facility. Use Left mouse button released as the event. Navigate in the message list until you locate Client. Select it...



The link editor shows the available client components

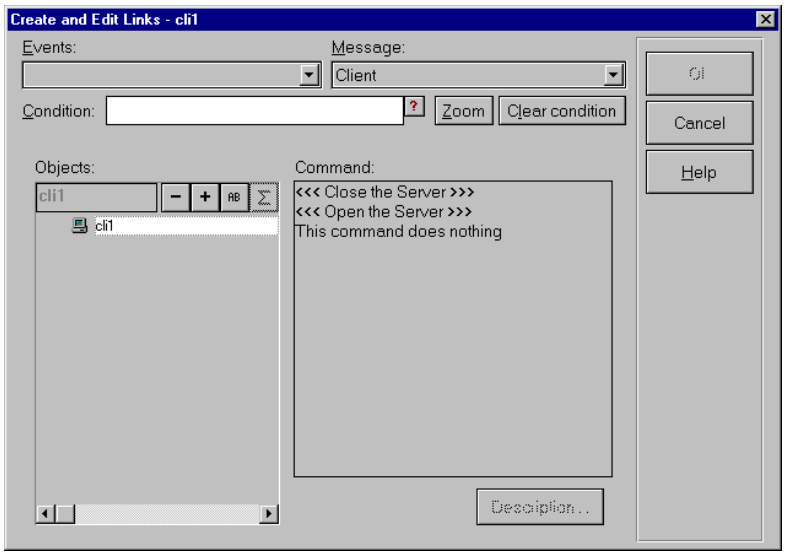
The list in the left side of this dialog box contains all client instances of the encapsulator component contained in this application.

You can place as many client instances of the encapsulator component into an application as needed. In fact, you can also place a server instance into an application which contains client instances. A mind-map application can contain a maximum of one server instance. A mind-map application can be called by, and can call, as many other subassemblies as are required. Thus, you can create networks of applications calling one another as needed.

Here comes the solution to the riddle…

Once you select the client instance CLI1 in the list on the left, it will query the server instance, residing in the other SRV1.MM file, to determine its functionality. Since this file is located on the local machine, the process will take a fraction of a second. Were the file contained on a computer on the other side of the globe, then the designated communication path would be established (i.e. TCP/ IP via the Internet) and the same process would be executed, obviously taking somewhat longer to complete.

The result of the interrogation will then be displayed in the list on the right side of the dialog box:



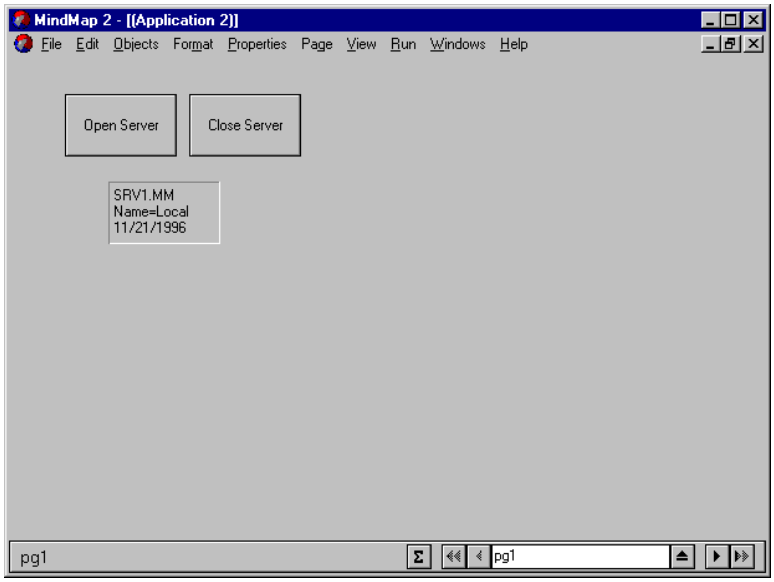
Select from the list of commands that the client component has retrieved from its corresponding server subassembly

Three commands are available. The first two are generated by default and are always available. The third command is the one we created in the server application. It appears exactly as we had input it.

Since we want to open the server application, select the second command <<<Open the Server>>> and acknowledge the dialog box.

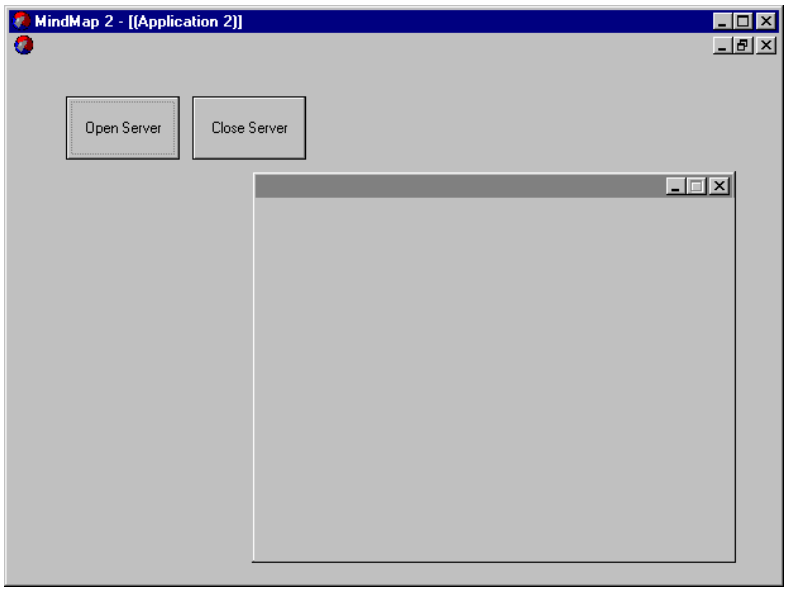
Now create a second command button and place more or less the same link on it, but instead of opening the server, generate the message <<<Close the Server>>>.

Your client application should look similar to this screen:



This is how the application looks

Again, save this little application and execute it. In run mode, click on the button you have designated to open the server application.



Clicking on the appropriate button in run mode opens the server application window

This is what the result should look like - again, more or less, depending on the labels you put on the buttons, and the dimensions you chose for the server window. To close the server application, click on the appropriate button in the client application.

Exchanging Events and Messages

Let us now take our little example one step further and have them interact with one another…

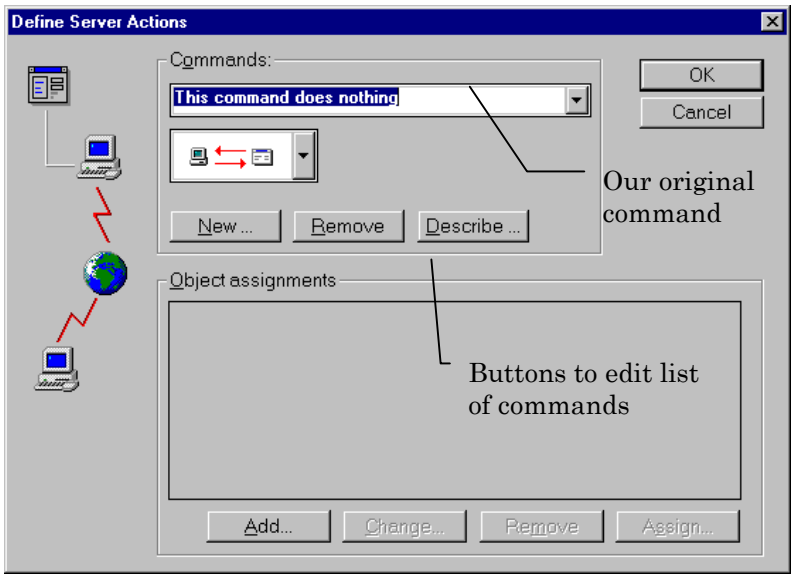
The first step will be to open the server application and place a rectangle in it. We will use the client to switch the fill color of the rectangle.

Open the server file SRV1.MM.

Place a rectangle in the top left corner of the screen. Then select the server instance and access its attribute toolbox. What we are looking to do is to add a new command to the server applications API (Application Programming Interface).

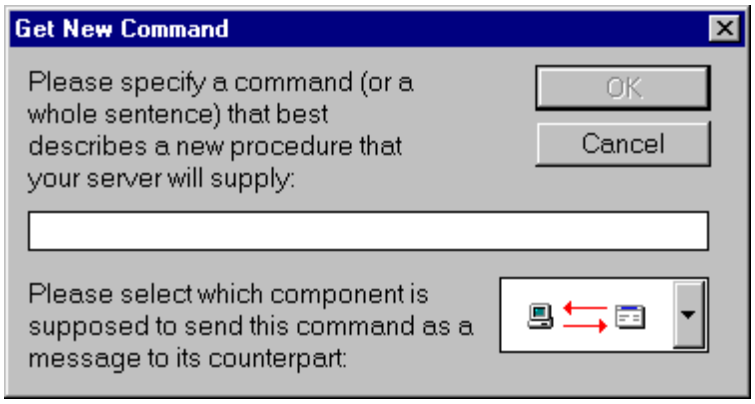
Attributes of a server component

Click on this icon and you will be presented with the following screen.



This dialog box is used to add additional functionality to your server subassembly

We will keep the dummy command and create a new one. Click on the New button.



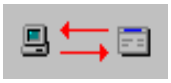
Enter the name of the new command and the direction of information flow

Again, enter a text to represent the command. In this case we will use something of the sort:

```
Change my color to red
```

The proper choice for the direction of a newly created command is essential for your server subassembly to be understood by a client. The following table may illustrate how the direction affects the registration of new events and messages:

Direction	The Server's View	The Client's View
	The server will find a new event that it can react on. Optional data will be transferred first, then the server component will receive the event.	The client will find a new message. Optional data that the client wants to transfer, must exist before the message is executed.
	The server will find a new entry in the appropriate link message. Executing this message causes optional data to be transferred to the client.	The client will be (asynchronously) notified that the server has executed the corresponding message. Any data has arrived in the associated components before the client is

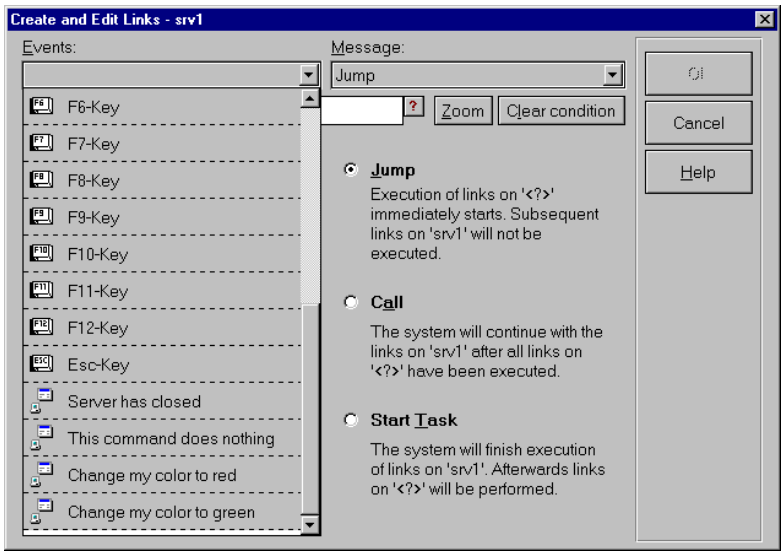
		notified.
	Both an event and a message are available. Data transfer works similar to the above descriptions.	Both an event and a message are available. Data transfer works similar to the above descriptions.

Leave the directional option set to the default of bi-directional, for the moment. Click on the OK button. Repeat this process to enter a command:

```
Change my color to blue
```

This was necessary to expose the functionality to the outside world. Now we need to define what the application will actually do when it receives either one of these commands.

Select the server instance and place a link on it. If you scroll down to the bottom of the event list, you will notice that four new events have been added to the list.

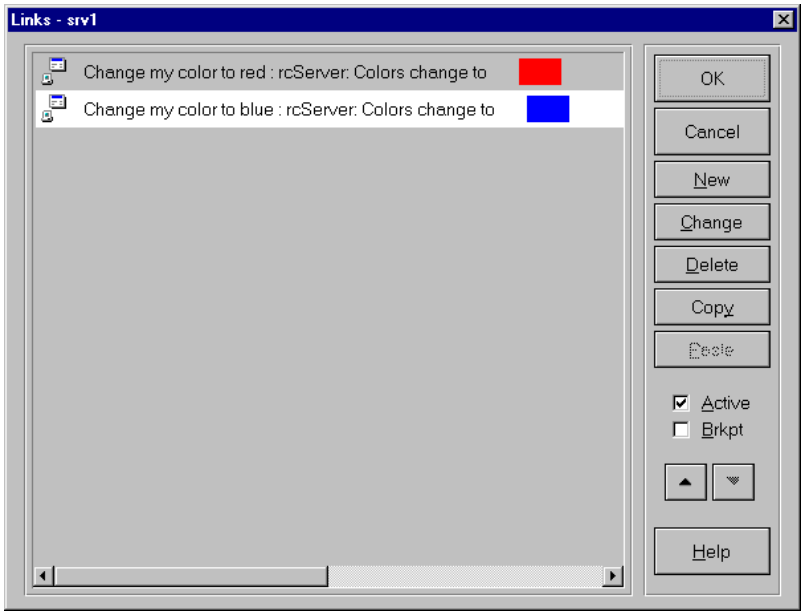


Special events introduced by the server component

Select the event:

'Change my color to red'

When this event is received by the server instance, it is to generate a message for the rectangle to change its color to red. Do the same for the blue color change.



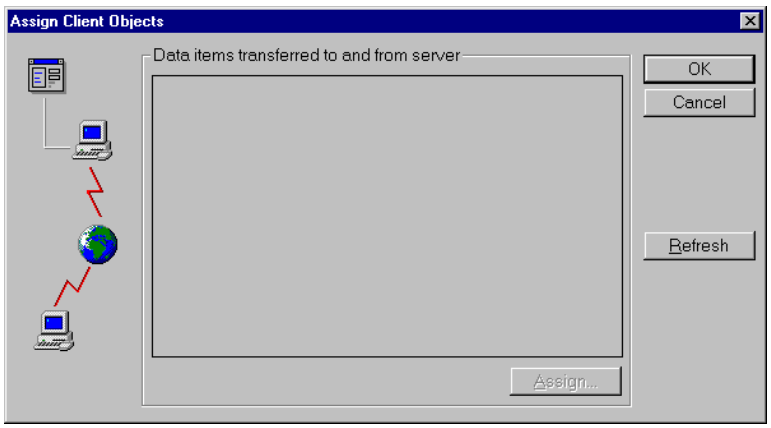
When the server receives an event, the appropriate link is executed

Return to the mindmap desktop, make sure that you have sized the screen so that it isn't in full screen before you save, and then save the application. Don't forget to save the application (SRV1.MM), since the client application will not know about the changes to the application, unless they have been saved. Now, close the file and load the client file CLI1.MM.

Begin the modifications to the client application by updating the connection between client and server. This is necessary to notify the client of new commands available at the server. In order to do this, select the client instance and activate its attribute toolbox. Only this time, select the icon for data exchange.

Attributes of a client component

You will be presented with the following screen:

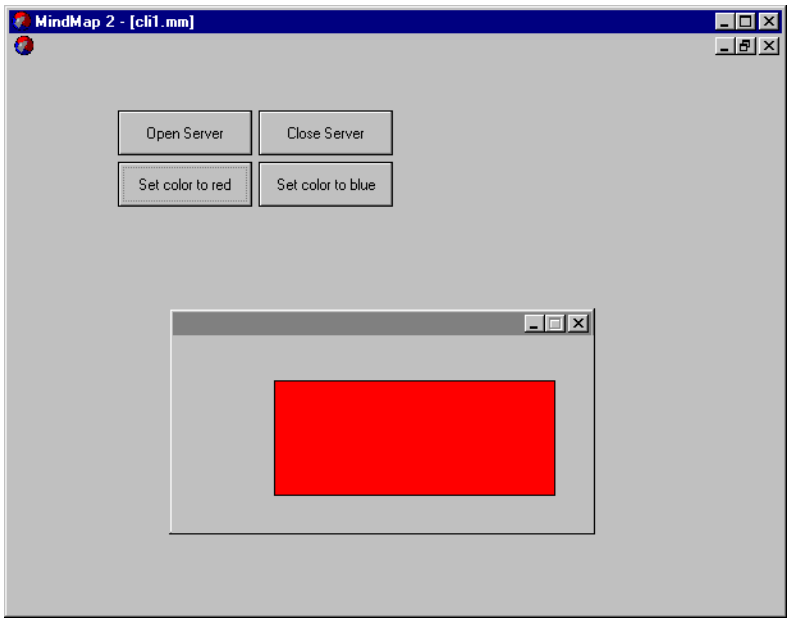


Connect server-supplied commands with components in your client-application

Click on the Refresh button and the client will call on the server once again to supply its updated API (Application Programming Interface).

Next, create a new command button which will be used to switch the color of the rectangle. Label it with an appropriate text. Continue by placing a link on the button which is to send a command to change the color to the server application - via the client instance and the server instance.

Save the application and execute it. If all went well, you should view a screen similar to this one…



The client application changed the color of a component in the server subassembly

Attributes

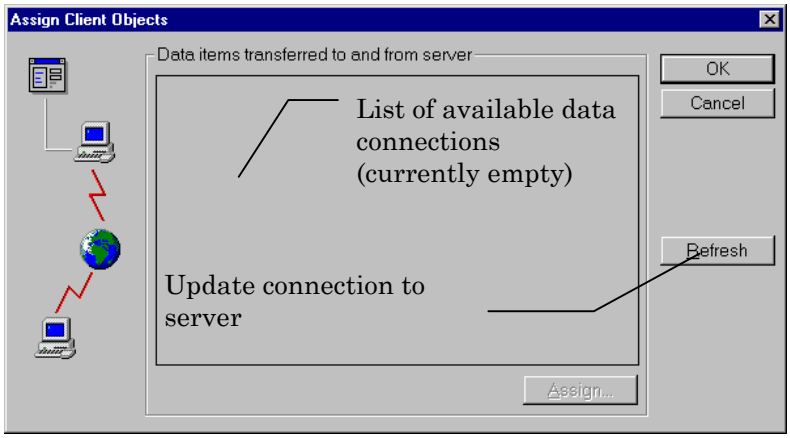
Client Instance Attributes

The client instance of the encapsulator has a total of four icons on the attribute box. The Linker and the Annotation feature are common to all other components and thus will not be described here.

Attribute of a client component

The Redefinition of Instance attribute allows you to redefine the properties of already placed instances. If the location of the corresponding server application has been altered, redefining is mandatory -- but it leaves predefined links intact as long as the server API has not changed since the application was initially created. This allows a server application to be built and tested locally on one machine and then moved to a remote machine without having to alter any underlying logic, other than reestablishing the connection (so that the client knows where to go looking for the server).

The Data Exchange attribute offers two different functions, albeit both are closely related:



The Refresh button makes sure that the list of available connections is up-to-date

The update will always be based on the contents of the file associated with the server subassembly. If the server subassembly is under construction, meaning the file is open, then only the status contained in the file can be accessed.

The first function allows you to update the connection to the connected server. Click on the Refresh button to have an update occur. If the server resides on the local machine, the update will finish immediately. If the server is stored on a remote machine, the communication connection must be established first and then the update will take place.

The second function allows you to assign data elements the server makes available to data elements contained in the client subassembly. A detailed description of this facility is available in the next section.

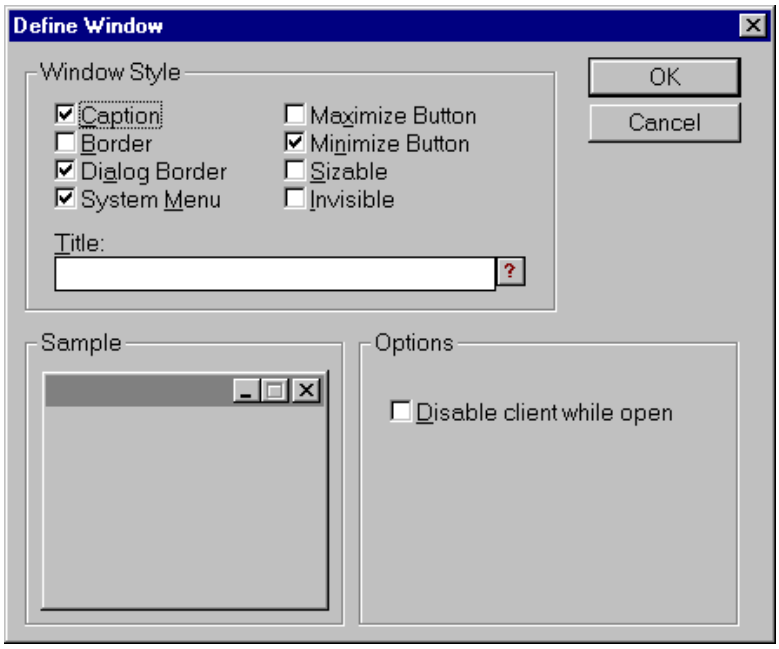
Server Instance Attributes

The server instance of the encapsulator component has three attributes, which are displayed on the attribute toolbox.

The annotation attribute is identical to the general annotation feature and is discussed in the general context. The same is true for the link facility.

Attribute of a client component

The icon in the top left corner of the toolbox offers various settings for the window itself. The icon immediately next to it relates to the exchange of data between a client and a server instance.



This dialog lets you modify the appearance of the server's window

The options available in regards to the window setting are:

- Caption: This allows you to enter a text which will appear as a caption on the window. The text is entered in the field entitled Title. While entering the caption it will appear on the little sample window in the bottom right section of this dialog box. Note that this field is evaluated through the parser. Therefore you may enter the name of another component.
- Border: Selecting this option will let a simple border display at run time. The mindmap icon will appear at the left corner of the caption bar.
- Dialog border: A dialog border will appear as a 3D beveled border. The mindmap icon will not appear on the caption bar.
- System menu: This option toggles the standard system menu on/off. This menu offers the user various options at

run time, including information regarding the mindmap application itself.

- Maximize button: Selecting this option puts the maximize button on the right side of the caption bar. This allows the user to expand the application to full screen.
- Minimize button: Selecting this option permits the user to minimize the mindmap application at run time.
- Sizable: If you select this option, the user will be able to change the size of the window at run time.
- Invisible: Selecting this option will keep server instance from displaying itself. It will remain functional, but cannot be seen. This option should not be selected if the server instance expects user interaction.

This dialog box also offers the option of disabling the client, as long as the server instance is open. This is generally used to avoid conflicts in terms of entering data and thereby changing the status of the client instance.

Exchanging Data between Client and Server

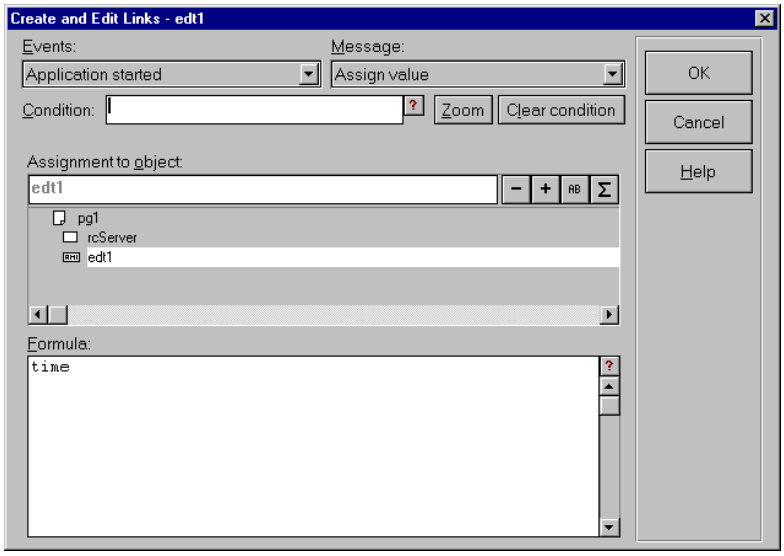
Just as data can be passed between components on one page of a mindmap application or among different pages in an application, so data can be exchanged between components residing in different .MM files. The underlying principle is again based on the encapsulator component. The called upon subassembly exposes its data to the outside world via the encapsulator component. The calling subassembly can see the available data components and can opt to connect these to its own data components.

Thus data components can either be local to a subassembly or global in the sense that they are visible to other applications. But this is a restricted global view. The server is always in command in regards to when the data is supplied. The client can send a message to the server requesting data. This will not actually cause the data to be transferred, though. The server can now, on its own, use this message as an event and generate a message back to the client with the requested data attached.

This is necessary to be able to eventually deal with asynchronous connections, in which an immediate response by the server can not always be guaranteed.

Begin by placing an input field in the client subassembly. Retain its default name (edt1).

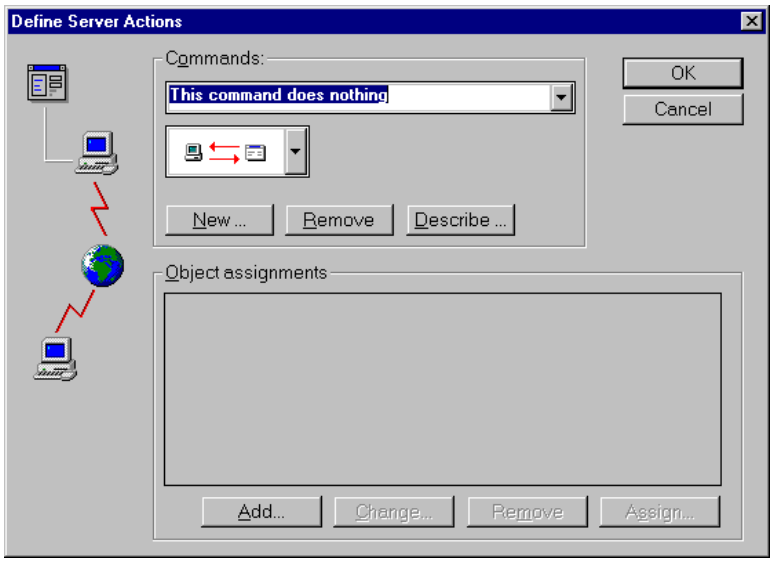
Next place an input field in the server subassembly . Assign a link to it which, upon the start of the application assigns the parser statement 'Time' to the input field. This will make the time display, once the application is put into run mode.



Let an input field show the current time when the server subassembly is loaded

Attribute of a client component

The next step is to expose the input field, containing the time in the server subassembly, to the client subassembly. In order to do this, access the attributes of the server instance of the encapsulator. Click on the data exchange icon and you will be presented with the following screen:

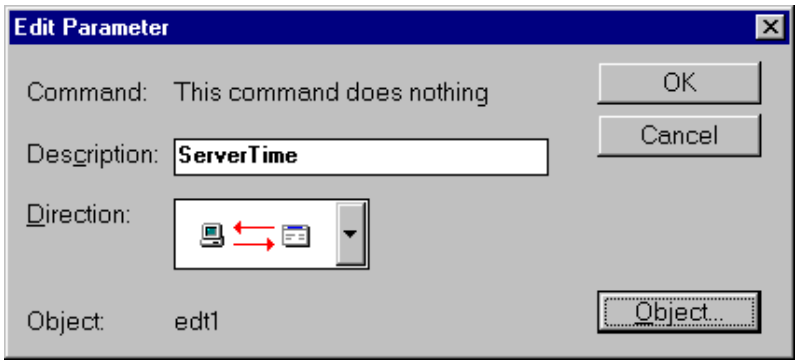


The server action dialog only shows the already defined command

In the bottom half of the screen you will see the list of object assignments. Here is where any components that are visible to the outside world would be listed. At the moment, none have been exposed.

First, you must select a message onto which the data will be attached when sent back to the requesting client. Let's use the dummy message, 'This command does nothing…'. Select it from the list at the top of the dialog box. As for direction, you can keep the bi-directional default setting for the moment. The next step is to attach the data to this message.

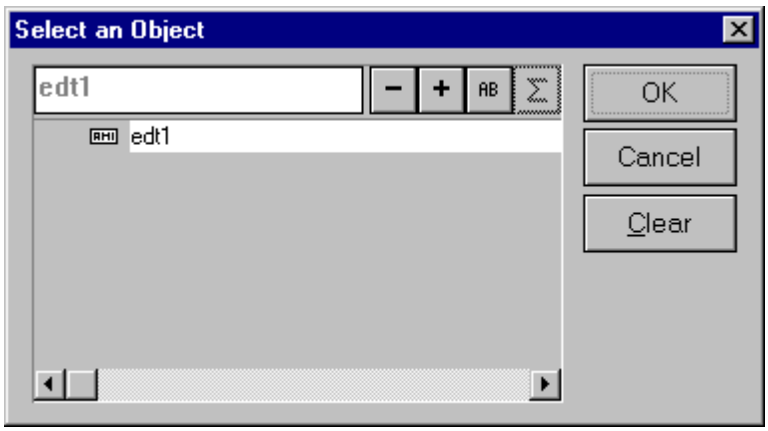
To do so, click on the button labeled Add. This will pop up the following dialog box.



Attach a component to a particular command and assign a name to it

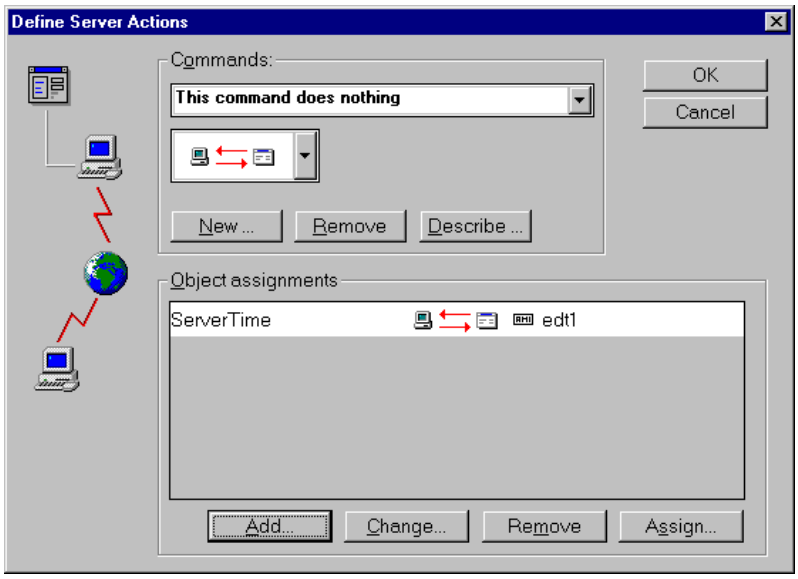
The name by which the input field is exposed to the outside world can be input in the Description field. In this case, it has been entered as ServerTime. Next we need to assign this (arbitrary) name to the actual component in the subassembly. Click on the button labeled Object. This will bring up the component list for this subassembly:

Data exchange is only possible between components which are capable of drag&drop, such as input fields, graphic components, data tables, list boxes, MCI, etc.



Select an object to be attached to the selected command

Select the input field and end by clicking on the OK button.



A new component has been assigned to a command

You can also assign by using the rubber band method.

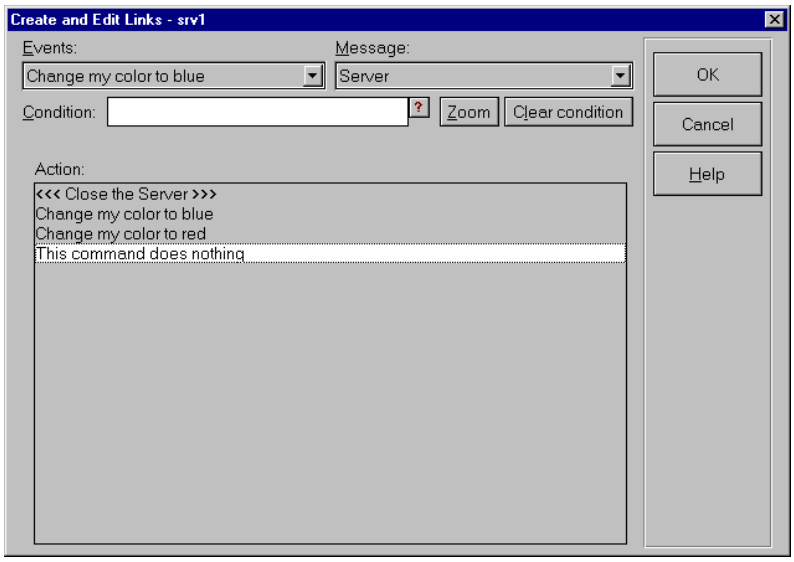
The next step is to associate an event with this message. In other words, the server subassembly will wait for a data request via an event. It will then reply to the event with its own message and attach the data to it. In our example, we have attached the data to the dummy message:

'This command does nothing...'

We will use the incoming event:

'Change my color to blue'

as the trigger event requesting data. To do this, we will simply define a new link.



The server component now supports more commands

The complete example is available in the directory SAMPLES\CLISRV as CLI2.MM and SRV2.MM.

When the server instance of the encapsulator receives the event:

'Change my color to blue'

it will respond with the message:

'This command does nothing...'

with the data attached to this command. The data is the input field named:

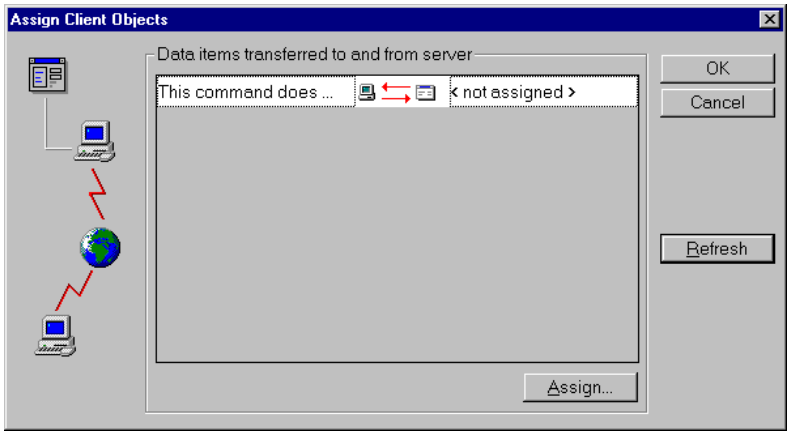
ServerTime

This is all that has to be done in the server subassembly. Now, save the subassembly into its file again.

The next step is to make the necessary assignment in the client side. Open the client file and access the attributes of the client instance of the encapsulator.

Attribute of a client component

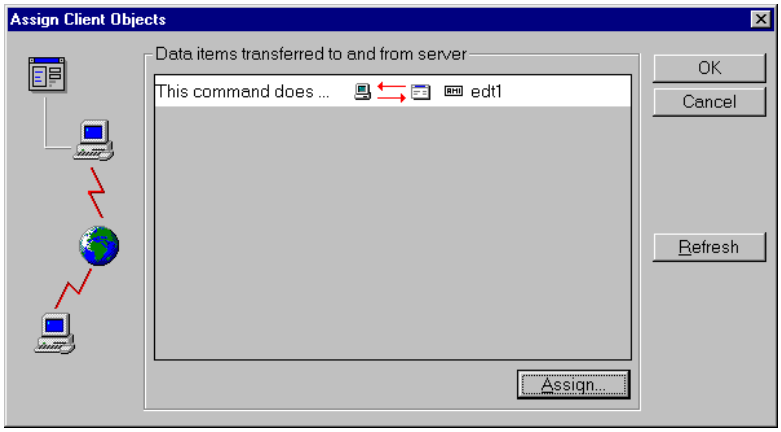
The first step is to update the client instance connection to the server connection. Click the refresh button on the Data Exchange dialog box. This should result in the display of the following screen:



Clicking on the Refresh button shows the newly defined assignment

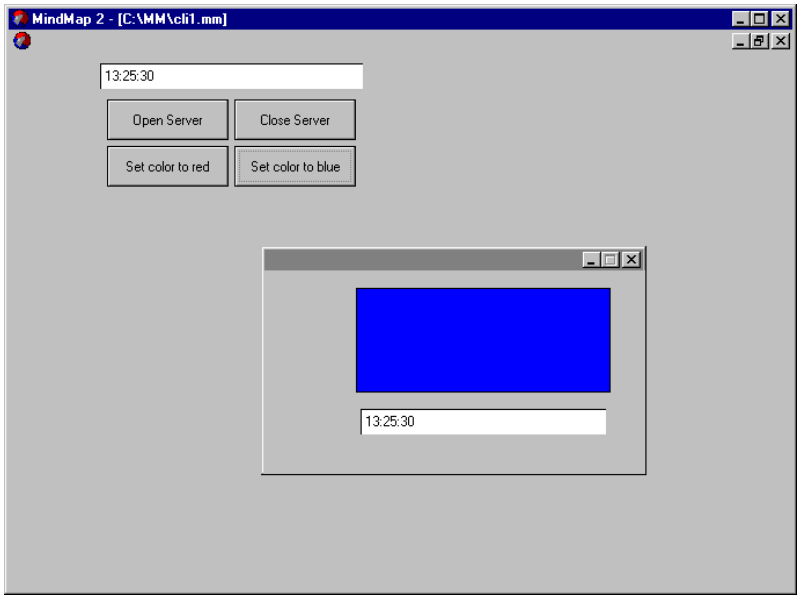
Now, the incoming data named ServerTime has to be assigned to a local component. Click on the Assign button. You will be presented with a list of all components in the client subassembly. Click on the input field named edt1.

You can also assign using the rubber band method.



The new parameter has been assigned to a component in the client application

Close this dialog box by clicking on the OK button. Save the client application and run it.



The server has filled the client's input field with the current time. The client requested this by sending a message

This has been a description of how data can be requested from a server. It is also possible to send data to a server. In such a case, it is not necessary to establish the 'call back' facility.

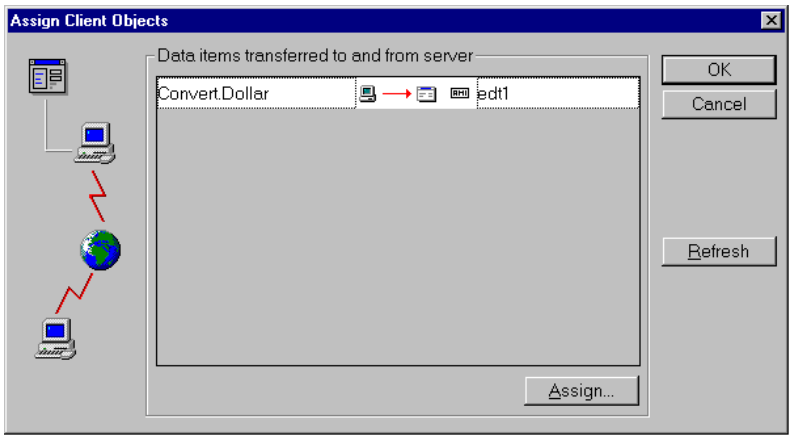
In exchanging data, you must assure that the data types of the two components correspond to one another. mindmap does not perform an automatic type conversion. This does not apply to the conversion of text-oriented information. For example, a numeric value will be automatically converted into a string if not otherwise supported by the target component. Especially note that the same rules apply as for the drag&drop link message.

Open a new file and place an input field. Then place a server instance of the encapsulator and define a dummy command. Assign the input field to the incoming global data component. Save the application into a file.

Open a new file representing the client side. Place an Open and a Close button and assign the appropriate links. Then place a Send button and an input field, the contents of which is to be transferred to the server.

Now place a link on the Send button so that the message you defined in the server will be sent to the server (via the client instance of the encapsulator).

The next step is to assign the input field to the global component, so that it can be transferred.

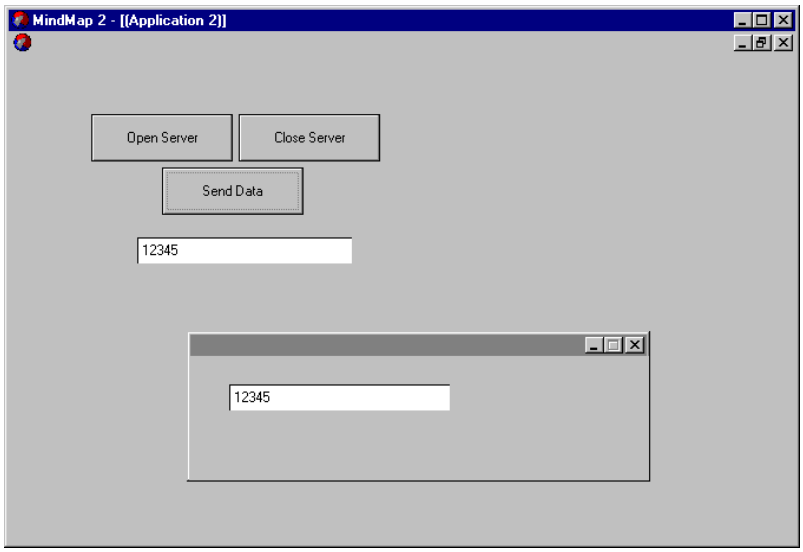


Another connection to a component

A client can never actually go out and get data from a server. It can only request the transfer of data from the server. The server must serve the data.

The internal component, named edt1, will be sent via the client instance of the encapsulator, as a component called Dollar.

Save this application in a file and run it.



The client application has sent information to the server subassembly

Special Events

Default Events / Server Instance

Every server instance always registers one standard event. It is:

- Server has Closed

This event can be used to send a message to attached clients, that the server has ceased operations.

Default Events / Client Instance

Every client instance registers two default events. These are:

- Server has Closed

- Server is Open.

User Defined Events

The user can define any number of events. In the context of the encapsulator component, they are referred to as commands. This is due to the fact that a command can be viewed as an event, as well as an action associated with a message, depending on the perspective.

Special Messages

If you have placed a client instance of the client/server component, two messages are automatically available. These are:

Open the Server

and

Close the Server

The first message will establish the physical and logical connection to the server instance and then open it. The second message will close the server instance and disconnect.

If you place a server instance, it will make the default message

Close the Server

available. This will terminate the connection based on an action taken by the server subassembly.

Parser Extensions

The client/server component does not register any specific extensions to the parser.

Limitations

The current version of the encapsulator component does not support streaming data. Thus, it cannot be used to transfer live video, sound, or animation data.

If you are running mindmap as a centralized server for a group of concurrently accessing clients, a number of limitations may apply.

Since every client connects to a new instance of its corresponding server application, many server windows will appear on the server console, if the server application is defined to run remotely. It is obvious that the number of open server windows is limited by the available memory and system resources. This restriction does not apply if a server application is transferred to the client computer and is executed there.

Be aware of the fact that other restrictions may apply due to limitations in the access to commonly used database resources, should your server application include a database component. (record locking, number of users, licenses, etc.).

Chapter 8 Making an Executable File

General Concepts

mindmap enables you to create a set of distribution media containing the mindmap application you wish to deploy to other computers. The distribution media will include all the mindmap files, associated modules such as databases, graphics, etc., and a setup routine.

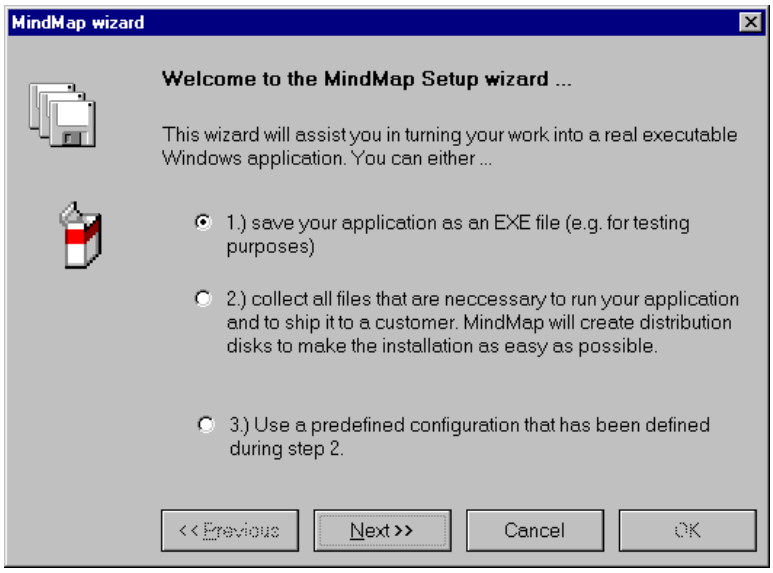
mindmap applications, when stored as .MM files, are editable by the developer as well as anyone else with a valid version of mindmap installed on their system. Under certain conditions, it is necessary to wrap an application up so that it can be run on machines other than those which have a copy of mindmap installed on them. In addition, it is often desirable to make the application executable-only, meaning that the user cannot modify the application.

Technically, mindmap takes an application (.MM file) and wraps it up, together with the MINDMAP.EXE itself, into a new file. It then automatically determines all necessary mindmap files required and offers you the opportunity to include any other files you wish to make available on the distribution media. The result of the process is an installable set of media, including a standalone, executable version of your application.

Test it First

When you are ready to deploy your application, it usually makes sense to first test the resulting EXE file. In order for you to do this, access the menu option **File | Make EXE file**

You will be presented with the following dialog box:

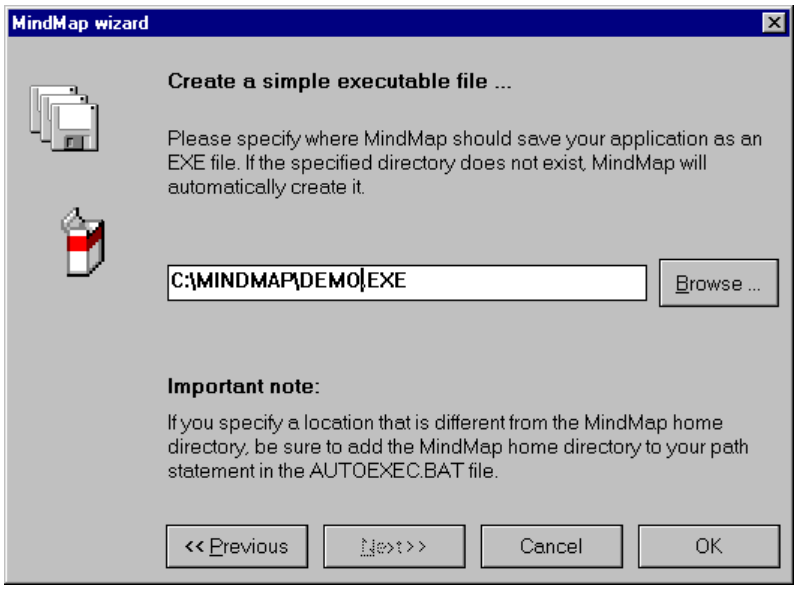


Create an executable file and test it first

The first option, which should be checked, gives you the opportunity to store your current .MM file as an .EXE file. This allows you to do a preliminary test of the application.

Press the Next>> button and mindmap will suggest a path and a file name - based on the name you have given to the current application (as an .MM file).

If you decided to store the application in a directory other that the default mindmap directory (where MINDMAP.EXE reside), please make sure that an entry in the PATH environment variable points to the mindmap directory. To accomplish this please use a text editor to open the AUTOEXEC.BAT file and add a line such as
PATH C:\MINDMAP
or modify an existing path definition.



Specify where to store the application

If you do not wish to store the .EXE file under the suggested name, either type in a new file name or browse to a new location on your hard drive.

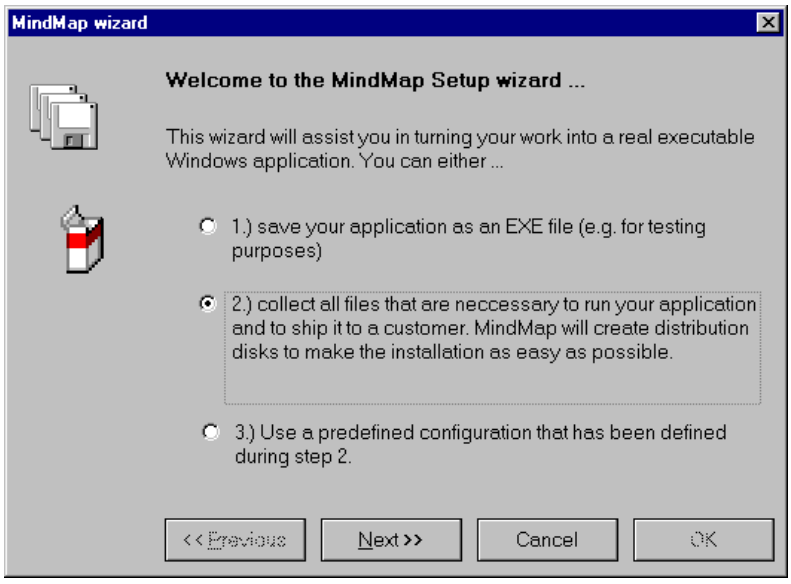
Now, exit mindmap and execute the newly created .EXE file, just as you would start any other application.

You will see that, once the application has started, you will only be able to minimize it or terminate it. You can no longer toggle into edit mode.

Building the EXE

Collecting the Files

If you actually intend to build a deployable EXE file (and associated files), then you should select the second option in the dialog box.



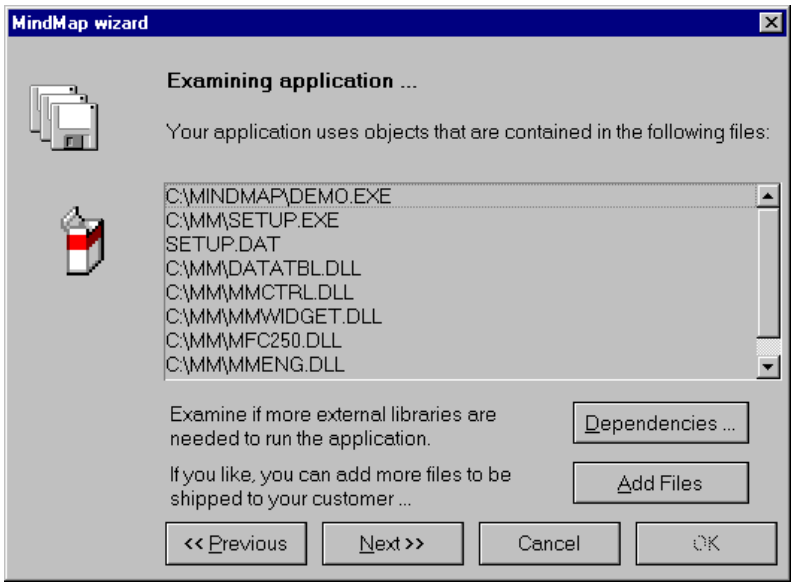
Build an installable set of disks from your application

When you instruct mindmap to collect all the necessary files, it will, by default, include a number of files which are required to run the system, regardless of the contents of your application. These include MINDMAP.EXE, the .MM file, and various libraries. Depending on which components were used in the application, additional files are added to the list.

Examining the Application

Default

The deployment routine will display the complete set of files required to make a standalone, executable version of your application.



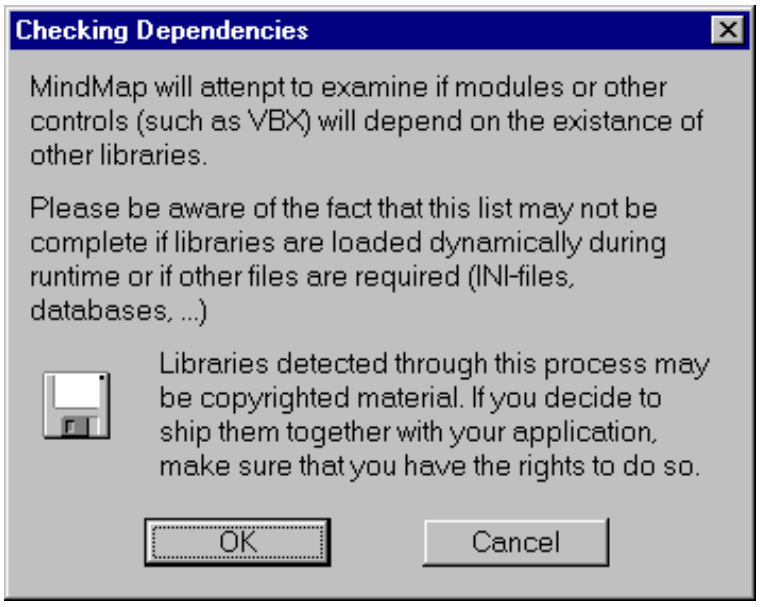
See which files are necessary to execute your application

All collected files are displayed in the list.

Dependencies

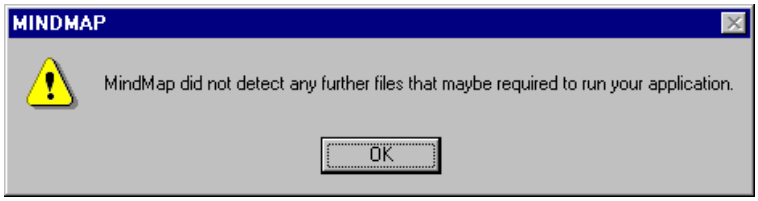
In some cases it might be necessary to manually augment this list with additional files. This might be the case for licensed materials (such as VBXs, third party libraries, etc.), or data files which are utilized in the application.

Click on the Dependencies button and you will see the following dialog box:

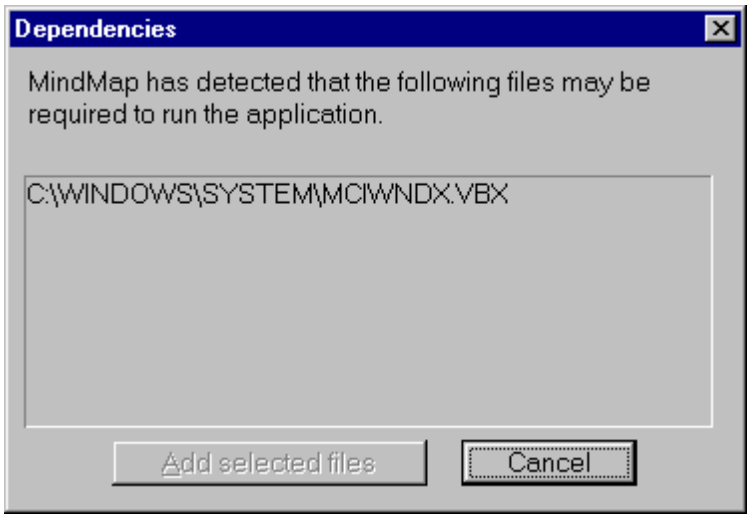


Make sure that no third party rights are violated by shipping certain components

If you have not included any external files in your application, mindmap will inform you accordingly and display the subsequent dialog box.



If on the other hand, mindmap has detected dependent files, these will be displayed in a separate dialog box.

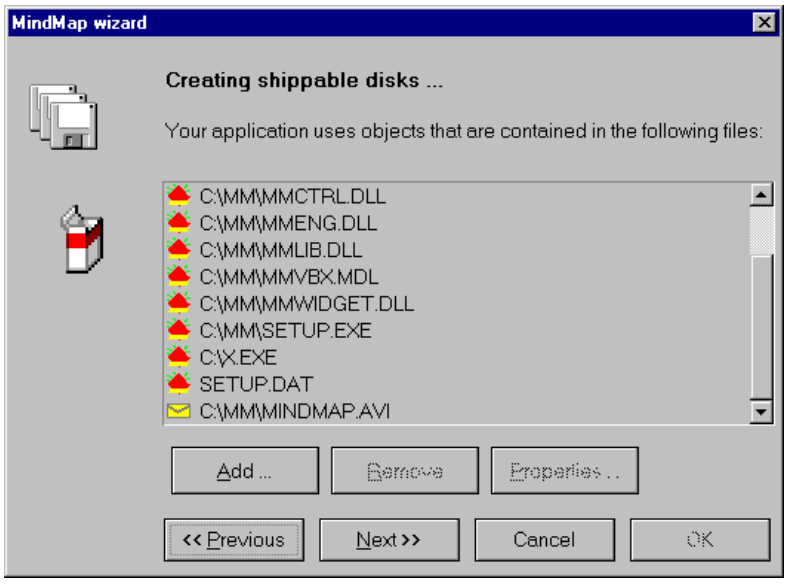


mindmap has found a VBX control that is used in your application

Please take note that such files might be licensed materials and their transfer may thus be governed by accompanying copyright agreements.

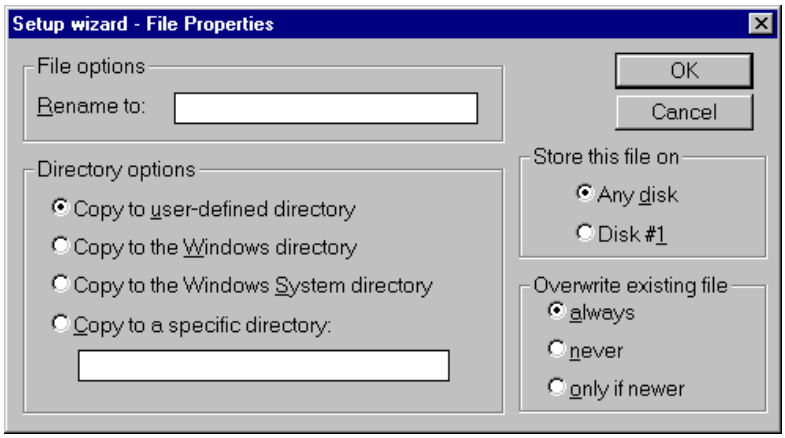
Adding Files

In some cases, you might wish to include additional files on the shippable disks. In this case, click on the Add button and navigate the resulting dialog box until you have located the file you wish to add. Once you have clicked on the appropriate button, control will be returned to the following dialog box, which will now display the newly added file.



Specify additional files to be shipped

You may now set some of the properties of the added file(s). In order to do this, click on the Properties button.



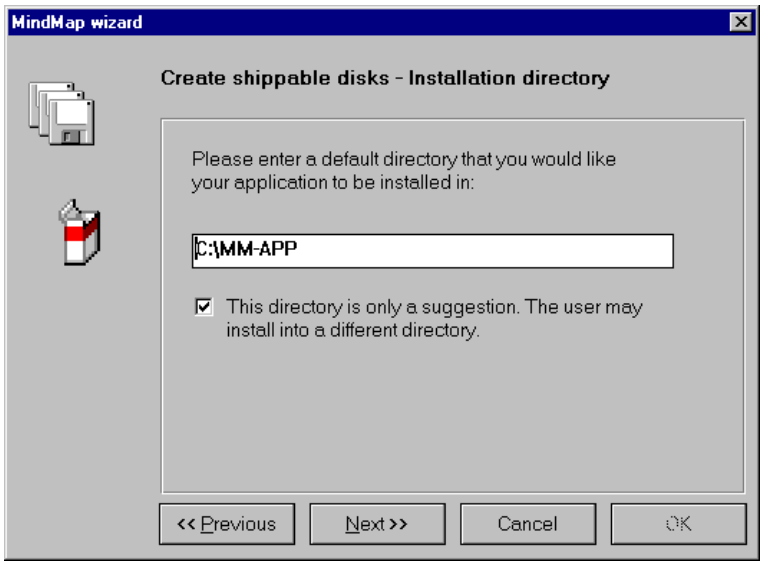
This dialog box gives you various options.

You might choose to rename the newly added file. You can also specify a definite location on the user file system into which

you want the file to be copied. You can also choose to overwrite the file, in case it might already exist on the target machine. You can use this feature to assure that the newest version of a particular file is always installed on the target machine. Also, note the option of selecting on which disk you wish the file to be copied.

Installation Directory

After having collected all the files, the deployment wizard will prompt you for a directory into which it is to install the application. The wizard suggests a directory \MM-APP on your default drive. You may either accept this suggestion or offer a different path.



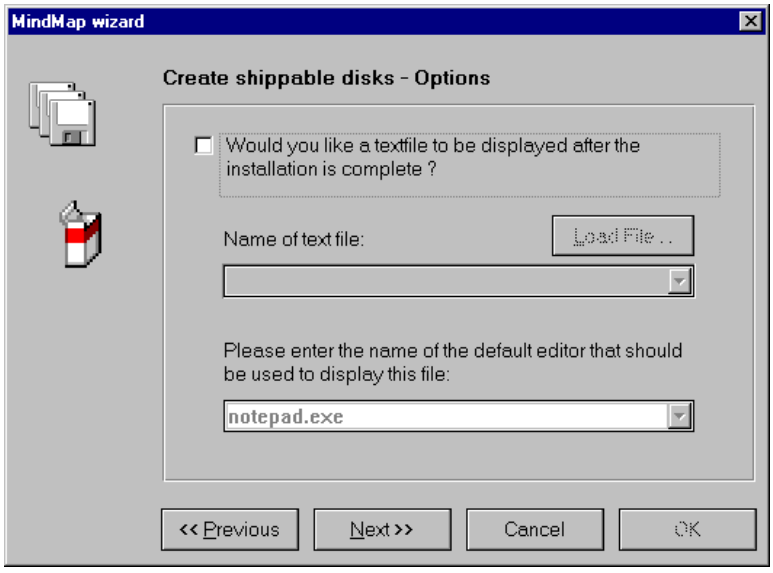
Enter a directory to which the application files will be installed

In either case, you may also offer the user the option of selecting his/her own directory of choice at the time of the installation.

Text File at Installation

Often it is desirable to display a text file after the installation process, informing your user about how to use your application or which other requirements he/she has to take into consideration. Such a text file might contain instructions regarding the installation, or it might include other information relevant at that time, such as copyright notices.

You can choose to display such a text. Specify the file and the viewer with which you wish to have it displayed:



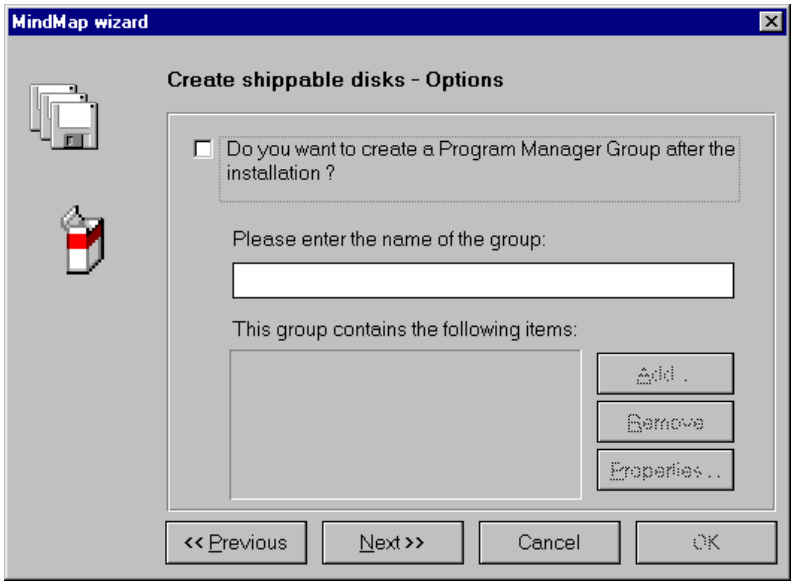
Is there anything your user has to take into account - present a text file

Please note that it is your responsibility to assure the availability of the specified viewer on the users' system. You might either make the assumption that the viewer is available by default (such as NOTEPAD.EXE) or you might choose to include it in the set of files you are shipping. Again, please take caution, that you do not neglect potential copyrights.

Program Manager

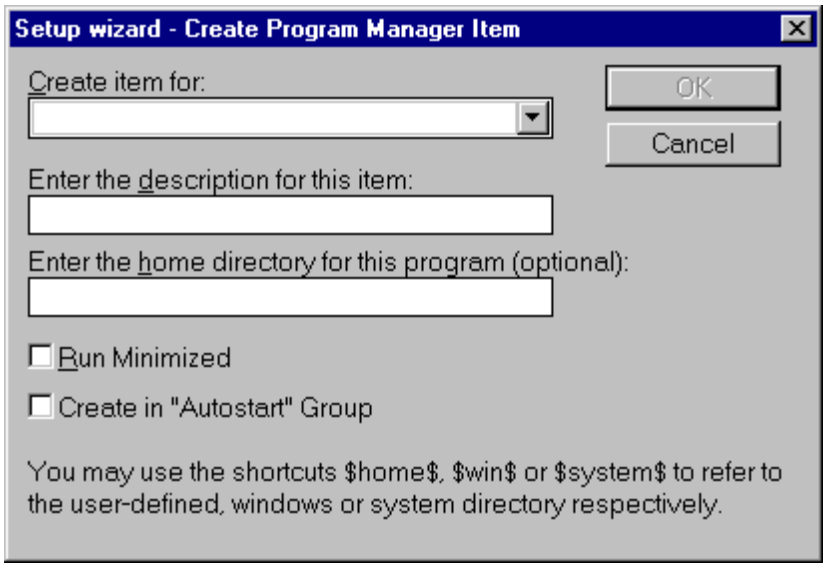
The next step in this process of creating deployable disks is that of specifying possible entries into the Program Manager.

If you wish to have the deployment wizard create such an entry, click on the check box. Next, enter the name which is to appear in the title bar of the program group. Follow this by selecting the files which are to be included in the program group.



Create some icons in the program manager so that the user can quickly start your application

Once you decide to include a file in the group, you will be presented with a dialog box permitting the selection of various properties:



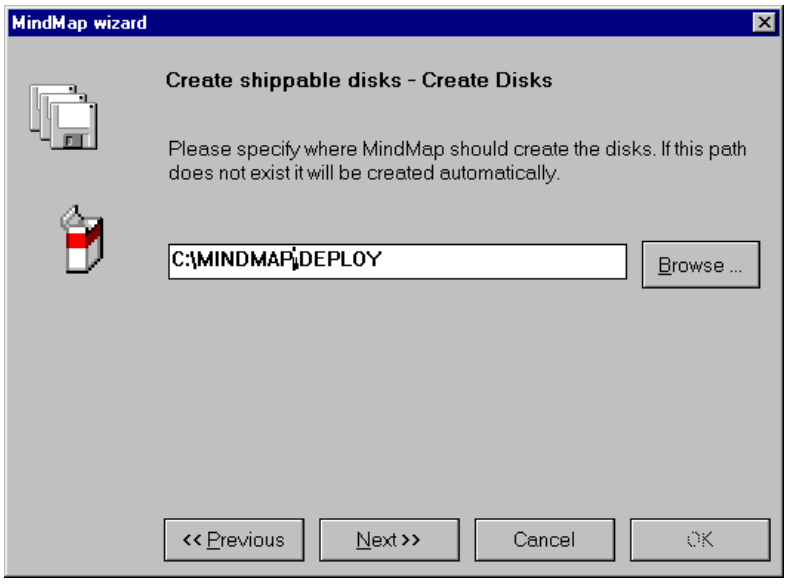
Specify which icons you want to create in the program manager window

The drop down list at the top of the dialog box permits you to specify for which of the files an item will be created. The list is filled with the files that are included in the shipping list created before. Next, enter a description for the file, followed by a specification of the home directory. Please note that you can enter relative paths, by adhering to the conventions listed at the bottom of the dialog box.

In addition, you can also specify that the application is to run when starting Windows and this, either in full screen mode (in which case it resides in foreground), or minimized as an icon on the desktop.

The Temporary Directory

In preparation for actually creating the disks, mindmap will temporarily designate a directory into which it will process the files on your system.



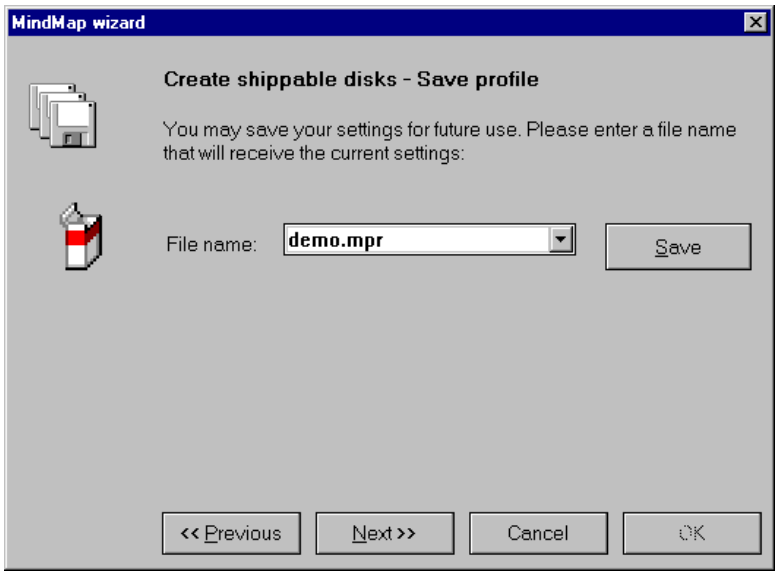
This is the location where your shippable disks will be created

The wizard will suggest creating a directory within the mindmap home directory. Again, feel free to override the suggestion and enter any other path.

This directory will contain the shippable disks which will be created in the next steps.

The Script File

If you intend to step through this identical setup procedure more than once, the wizard offers an option by which the individual steps can be stored in a script file.



Optionally save your settings in a script file for future use

Note that the script files are binary and cannot be processed in any manner outside of mindmap.

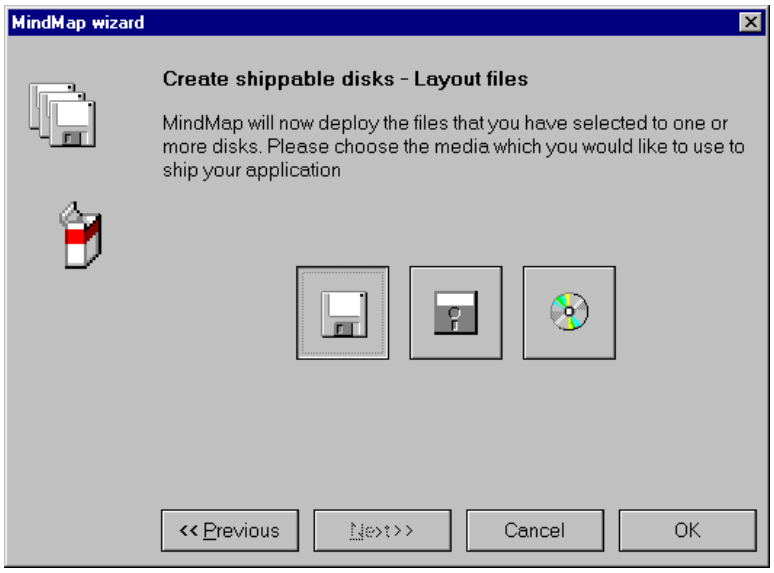
Again, the suggested name for the file is based on the name given to the application, as such. Feel free to override and enter a different file name.

If you do choose to save the script file, the wizard will recognize the existence of such a file when it is next invoked and prompt you to select the appropriate script file to utilize.

Media Selection

The final step in preparing for the actual creation is the selection of the media on which the application is to be deployed. This decision is necessary to determine the number of media required.

The wizard offers one of three selections:



The last step: which media do you want to have created

The wizard will create the appropriate number of directories. These directories are named DISK1, DISK2, etc. and the deployment wizard will assure that no more files are copied into these directories than will fit on the disk. It will create only a single directory if the option CD-ROM was chosen.

During the compression process, a directory named COMPRESS will automatically be created. This directory contains the compressed versions of the files to be shipped. This directory will not be removed after the wizard has completed. Therefore subsequent creations of shippable disks will be sped up. Anyway, feel free to remove this directory any time, if you want to free the disk space.

Once the media has been selected, the wizard will compress all the selected files, and rearrange them such that a maximum number of bytes will fit onto each piece of distribution media. Then, it will copy these designated files into the directories corresponding to each media and report upon completion of the task.

mindmap lets you know where the disks have been created

If you did not select to create a CD-ROM image, you may now choose to use the wizard to copy the contents of the displayed directories to the media (by actually specifying the drive). Alternatively, you can leave the wizard and transfer the contents of the directories to the distribution media using any other utility, at this time or in the future, as you wish.

Chapter 9 Printer Layout

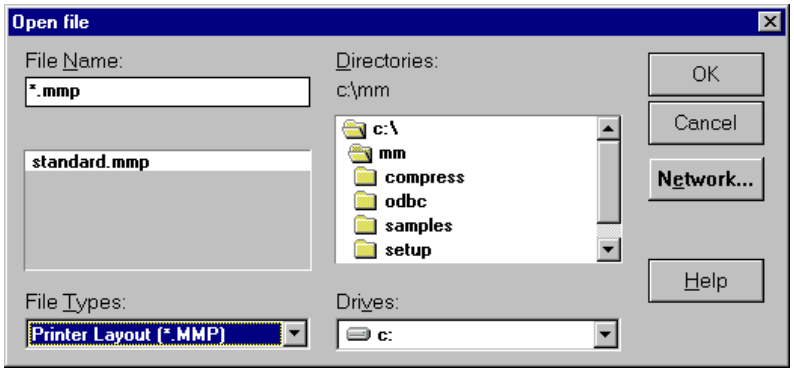
General Concept

mindmap offers the ability to print the internal documentation of an application built with mindmap. This is helpful, if other persons are to review, modify, or maintain the application. This documentation is continuously generated and can be output in various formats. mindmap is shipped with a collection of templates for printing the documentation. These templates are contained in a file named STANDARD.MMP (MMP = mindmap Printer Layout).

The standard templates themselves are, technically, little mindmap applications and have been created using mindmap's own tools.

Conceptually, a printer layout is a blank sheet of paper onto which mindmap components can be placed (graphical primitives, graphic file imports, text, and printer components). You can then access most attributes of the components you place on the sheet, just as though they had been placed on a mindmap page. By defining the various components on the page, mindmap is instructed how to eventually display the documentation.

Let's start out by loading the standard templates and viewing how they have been constructed. Start by opening the appropriate file using the menu option **File | Open**. This will result in the following screen:



Open the predefined printer template

In the lower left hand corner, you will see a drop down list which offers a selection of various file types. Select the entry:

Printer Layout (*.MMP)

The list in the top left hand corner of the dialog box will subsequently display all files in the current directory corresponding to the .MMP extension. Now open the file:

STANDARD.MMP

by either double-clicking on it or by selecting it and then clicking on the OK button. This will open the printer layout file.

On the left side of the mindmap page, you will now see a white rectangle. This rectangle represents a sheet of paper. A rectangle is also visible on the page, along with a horizontal line.

You will also notice that the mindmap page itself has a name - mindmap Page (graphical). If you leaf to the right, you will observe additional pages. There are five pages in this file:

- mindmap Page (graphical)
- Page Overview and Links
- Component List
- Components, Links and Annotations
- Components and Annotations

Each page has a somewhat different layout.

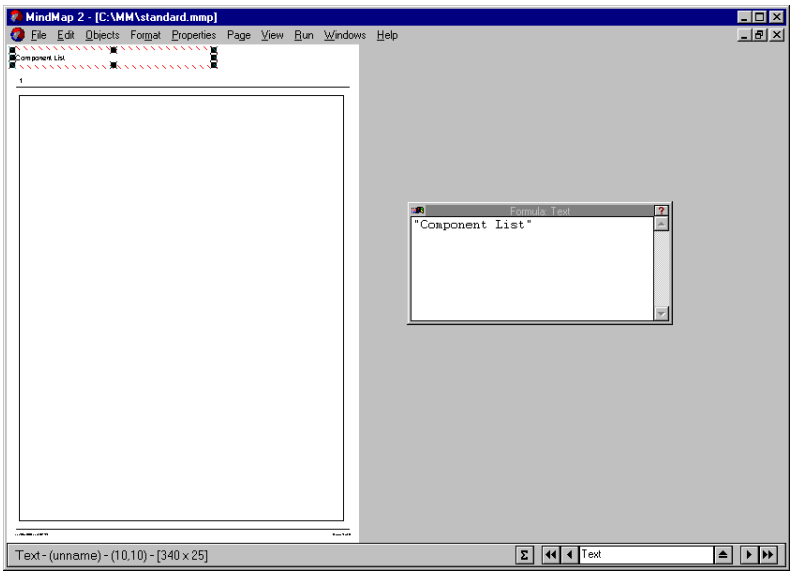
The template mindmap Page (graphical) is used to print a mindmap page, as a user would see it in run mode. The Page Overview and Links template will produce a representation of the mindmap page and all links placed on the page. The Component List template lists all components on a page. The Components, Links and Annotations template displays all components on a page, along with their links and any annotations. The last template outputs the components and their annotations in a page orientation.

Changing an Existing Template

The first step in understanding and using the Printer Layout facility is to change some parts of an existing template.

Assuming that the template file STANDARD.MMP is still open, scroll to the third page, which is entitled Component List.

You should see the following screen:



View the formula editor for a text component in a printer template

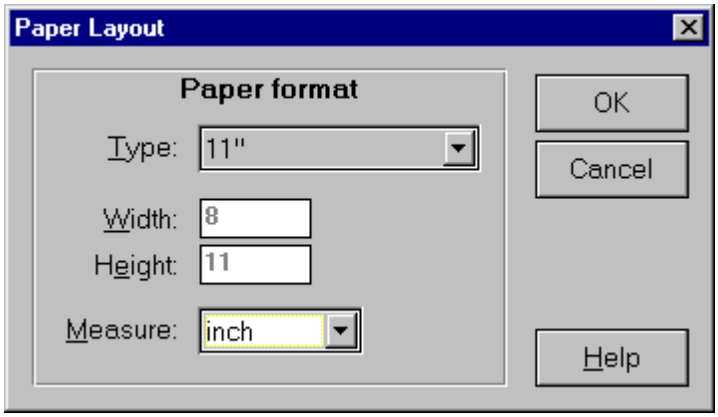
Selected in the top left corner of the page, you can see the page header, which in this case is Component List. This is a simple text component and it has received its value by an entry in the parser field. Please note that string entries must always be enclosed in "…". Use the cursor to select the other components on this template page, these being horizontal lines (at top and bottom respectively). A footer which contains two text components, one of which displays date and time of printing, the other a page number reference.

Let's add your name to the header:

1. Click on the text component on the toolbox or select it via the menu. Place the cursor on the top right hand section of the template and drag it open.
2. Next, assuring that the text component is still selected, click on the parser button on the status bar on the bottom of the mindmap page. This will pop open the parser window.
3. Assuring that the newly placed text component is still selected (it should have a red hatched border), click inside the parser window.
4. Now type in your name, making sure that your name is enclosed in double quotes. As soon as you place the focus outside the parser window, the newly created text component will reflect the text entry.
5. Follow this by selecting a font for your name. Again, this is exactly the same procedure as though you were to select a font for any other component placed on a mindmap page. Bring up the attribute toolbox on the text component containing your name, and proceed to make the font selection.
6. Finally, save these changes back to the STANDARD.MMP file.

Using the standard mindmap procedures for placing components and defining their attributes, you can now change the appearance of any of the existing templates.

Before you start to use your templates though, you might want to check that the correct page settings have been made. To do this, double-click on any section of the template, this means you shouldn't be clicking on any of the components, since you don't want to set their attributes. If you select the template page, this will result in the following dialog box:

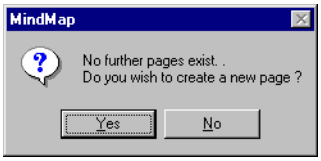


Select a paper size for the printout

Use this dialog box to make the necessary adjustments. In case you did make some changes in this step, save the file again.

Adding a New Template

The existing template file can always be expanded to include new, user-defined templates. In order to do this, you have to follow the steps outlined below:

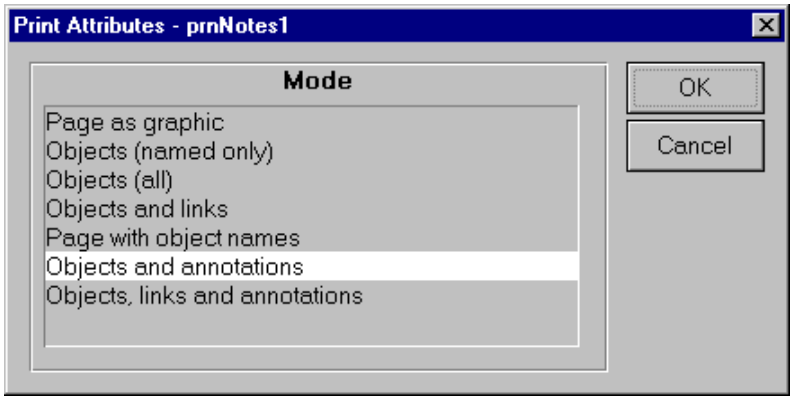


Press Yes to create a new page in an application

- Create a new mindmap page by either using the menu option **Page | New Page**, or by going to page number 5 in the standard file, and then simply attempting to go one more page. You will be prompted, in which case you should answer with yes.
- You will be presented with an empty 'sheet of paper' on the left side of the mindmap page. This is the basis for the new documentation template.
- Now you need to place a new type of a component on the template, which is the container for the aspects of mindmap you wish to print. This component is called Printer Object. You can only access it via the menu option **Objects | Printer Object** (described on page 144). It has no representation on the toolbox, as it is infrequently used. Select it from the menu and drag it open on the template.

Attributes of the printer component

- Once you have done this, a rectangle will represent the new component. Now you need to specify what it is to actually contain -- what you want to fill it with. To do this, activate its attribute toolbox. Click on its specific attributes and a dialog box offering a selection of options will appear. Pick one of the available options and acknowledge the selection by clicking on the OK button:



Various output options are offered

- Next, you can start to enhance the general appearance of the page. You can define a header and a footer, add textual comments, etc.
- Finally, give the mindmap page a name. This requires that none of the components be selected. This name will become the template's name.
- Save the file to STANDARD.MMP.

Close the file completely and then reopen it. At the end of the file, you should now have the previously created template.

Open a mindmap application (.MM) and go to **File | Print Preview** menu (see page 38 for more information). The list of templates should now display the newly created template, which should be at the bottom of the list. Print the documentation for the loaded .MM file and see how it looks.

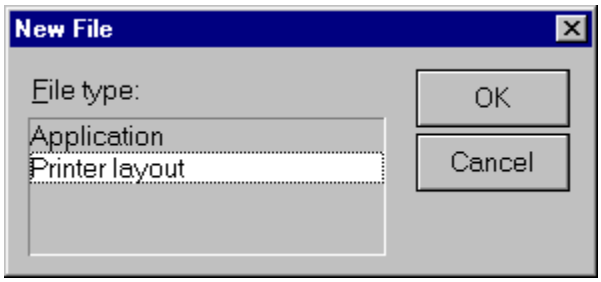
If you don't like the appearance of the results, reload the STANDARD.MMP file. Go to the page in the file containing

your new template, make the adjustments and save the template file, again.

Creating a New Template File

We have just discussed how the existing template file can be expanded. In addition to this, you can also create any number of new printer layout files. mindmap will search for these (files with the .MMP extension) in its working directory and load them along with the STANDARD.MMP file. All templates will then appear in the selection list, when you start to print the documentation.

To create such a template file, select the menu option **File | New**. This will result in the following dialog box:



When creating a new application you are prompted to select the desired type

Select the second option, Printer layout. This will open a new mindmap application file, which has a 'sheet of paper' on each mindmap page. Now, proceed to place the Printer Objects on the pages, adding any additional components you desire.

Don't forget to name each page, so that you can recognize them later in the selection list.

When all has been done, save the file with the extension .MMP. Make sure that you are saving it to the working directory, so that mindmap will automatically load it.

Chapter 10 The Parser

Any mindmap component may contain a formula that produces a result. A formula may be associated with a component or embedded in the component. In order to embed (you), you invoke the Formula Editor and define the formula. If you wish to associate a formula with an object, this is accomplished in any of the many Link dialog boxes. Input fields that are capable of dealing with formulas can be recognized by a small command button in their upper right corner which displays a question mark.

You may create formulas using:

- constants or values
- mathematical, comparison, and logical operators
- references to other components
- built-in routines called functions

You may combine these elements into an expression. For example, a formula for a field 'Invoice Total' could be:

```
(Subtotal * 1.12) + Shipping
```

Overview

Constants

Constants are values in formulas that do not change:

Examples:

- `Shipping * 1.12` ; 1.12 is a constant
- `"Due " + date` ; 'Due' is a text constant

When you include a text constant in a formula, you must enclose the text constant with either single or double quotation

marks. To include quotation marks within a text constant, you must use backslash quotation marks:

Example:

- `"Date \"Second Notice\" Due: "`
  ; results in Date 'Second Notice' Due:

Operators

Operators are symbols indicating how to combine two or more expressions. They include the standard arithmetic operators (+, -, *, /, ^, and %) and some special operators discussed later:

- `SubTotal * 1.12` ; * is the multiplication operator
- `"Date due "+ strdate`
  ; + combines text values

Component References

Component names may be used in formulas to include the contents of the component in the formula. When mindmap evaluates the formula, the contents of the component replaces the name of the component.

Examples:

- `SubTotal * 1.12` ; SubTotal is a component name
- `SubTotal / AmountOrders`
  ; AmountOrders is a component

Functions

Functions perform frequently used calculations (or operations) and are generally followed by parameters (values or references to components). Functions are described in detail later.

Examples:

- `date` ; Supplies the current system date
- `cos(edt1)` ; returns the cosine value of edt1

Expressions

An expression is a logical sequence of functions, values, constants, and operators working together to return a single result. Expressions are like clauses and phrases in a sentence.

Examples:

- `(SubTotal * 1.12) + Shipping`
- `substr(Title, 2, 5) + Name`

Using appropriate operators and functions, you may construct various kinds of values within a formula. However, a formula may have only one type of result:

- string
- numeric
- date
- time

The internal representation of strings is ASCIIZ. Call-by-Name references to strings are guaranteed to supply a buffer size of 4096 bytes, including the terminating null character. Numeric values are represented as doubles. Dates are represented as unsigned long (32-bit) values, interpreted as Julian date. Time is an unsigned long (32-bit) value, interpreted as seconds.

You must take care in constructing expressions with regards to the various types of data used. Using incorrect types might lead to unexpected results. For numerical errors occurring in mathematical functions of the parser, the value expected to be returned from the function is replaced with 0, if the function reports an error. An error message is appended to the system log file, as well as to the parser window, if applicable (see mindmap menu **File | Preferences | Formula** on page 46).

Currently, the math runtime library supports the following error messages:

- 'Numerical overflow in function fn(arg)'
- 'Invalid argument range in function fn(arg)'
- 'Argument singularity in function fn(arg)'
- 'Partial loss of precision in function fn(arg)'

- 'Total loss of precision in function fn(arg)'

where fn(arg) is replaced with the function name and argument that caused the error.

Please note, that for object-registered functions, the parser only defines the requested data type for passed arguments, whereas the data type returned by the function may depend on the specific implementation. For example, the function GetProp, registered by the VBX object, returns a data type depending on the name of the property. Passing this value directly to a function might cause problems, if the function expects a different type. These problems have been avoided by having mindmap automatically perform a reasonable conversion before passing the argument.

mindmap automatically converts into appropriate types and appends a warning to the system log. This applies also to operators that combine different types. Almost always, mindmap attempts to concatenate the factors as strings.

Operators

mindmap supplies various types of operators to work with expressions:

- mathematical
- comparison
- logical
- text

By definition, operators are placed between two expression elements. The following tables show all the mindmap operators and indicate how two values are combined:

Mathematical Operators

Symbol	Name	Definition and Example
+	Plus	Add two values `2 + 2`

		`SubTotal + Shipping`

-	Minus	Subtracts second value from first value
		`2 - 2`
		`SubTotal - Discount`

*	Multiply	Multiplies two values
		`2 * 2`
		`SubTotal * SalesTax`

/	Divide	Divides the first value by the second
		`Meters/100`
		`500/2.5`
		An error message is generated if the divisor is zero.

Mod
%
	Modulus	Calculates the remainder of a division
		`5 mod 3` ;equals 2
		`6 mod 3` ;equals 0
		The percent symbol can be used in place of 'mod'.
		`5 % 3`
		`6 % 3`

^	Power	`10 ^ 2` ; equals 100 `2 ^ 5` ; equals 32 The result of this operation is 1 if both expressions are zero.
( )	Precedence	mindmap evaluates formulas from left to right, performing multiplications and divisions before additions and subtractions. Using parentheses allows you to change the order. mindmap evaluates expressions between parentheses first. `3 + 5 * 5` ;equals 28 `(3 + 5) * 5` ;equals 40

Comparison Operators

These operators compare two values and return either true or false. These expressions are often referred to as Boolean expressions. Arithmetically, a result of true equals 1 and a false equals 0.

Symbol	Name	Definition and Example
= ==	Equals	True if both elements are equal `23 = 29` ; false `27 = 27` ; true
<> !=	Not equals	True if both elements are not equal `23 <> 29` ; true `27 <> 27` ; false
>	Greater than	True if the value on the left of the operator exceeds the one on the right `23 > 29` ; false `23 > 23` ; false `23 > 19` ; true
<	Less Than	True if the value on the left of the operator is less than the value on the right `23 < 29` ; true `23 < 23` ; false `23 < 19` ; false
>=	Greater Than or Equal To	True if the value on the left is greater than or equal to the value on the right `23 >= 19` ; true `23 >= 23` ; true `23 >= 29` ; false
<=	Less Than or Equal to	True if the value on the left is less than or equal to the value on the right

		`23 <= 19` ; false `23 <= 23` ; true `23 <= 29` ; true

Logical Operators

Logical operators can build compound conditions into a formula. Sometimes, two or more conditions must be met before you choose a particular method of calculation. Logical operators enable you to describe such combinations of conditions.

Symbol	Definition and Example
AND	True only if both elements are true 'true' AND 'true' equals 'true' 'true' AND 'false' equals 'false' 'false' AND 'true' equals 'false' 'false' AND 'false' equals 'false'
OR	True if either of the two values is true 'true' OR 'true' equals 'true' 'true' OR 'false' equals 'true' 'false" OR 'true' equals 'true' 'false' OR 'false' equals 'false'
XOR	True if either, but not both values, are true 'true' OR 'true' equals 'false' 'true' OR 'false' equals 'true' 'false' OR 'true' equals 'true' 'false' OR 'false' equals 'false'
NOT	Changes the value of the subsequent Boolean operation from true to false or from false to true NOT 5 = 3 equals 'true'

AND and OR are used to combine two expressions. NOT is only used on a single expression.

The result of a logical operation is treated as a numeric value having the possible values 0 or 1. Be aware that calculations with the results of logical operations are possible.

Text Operators

You may use text operators to indicate text constants in your formula or to combine two or more expressions into one expression.

Symbol	Name	Definition and Examples
+	Concatenate	Appends the text string on the right to the end of the text string on the left `"abc" + "def"` ; equals 'abcdef'

Special Symbols to be used in text constants

The following symbols are valid only within string constants (such as text enclosed in single or double quotation marks) and may be used to denote special characters in the text string.

Symbol	Name	Definition and Examples
" " ' '	Text Constant	Marks the beginning and the end of characters to be considered a text constant. Quotes without text between them indicate a blank space. If you enter text into a formula without using quotes, mindmap interprets the text as a component name or as a function.

		"edt1" ; a text string containing the four characters edt1 "mindmap" ; a text string containing the seven characters mindmap edt1 ; refers to a component with the name edt1
\\	Backslash	To represent a backslash in a text string, one has to put a backslash in front of it. 'c:\\mindmap\\samples' results in c:\mindmap\samples
\'	Double Quotation mark	To represent double quotation marks in a text string, a backslash has to be set in front of each quote. "Tool \"mindmap\"" results in Tool 'mindmap' Alternatively, the following form may be used: "Tool 'mindmap'" results in Tool 'mindmap'
\'	Single Quotation mark	To represent single quotation marks in a text string, a backslash has to be set in front of each quote. "Tool \"mindmap\"" results in Tool 'mindmap' Alternatively, the following form may be used: "Tool "mindmap"" results in Tool 'mindmap'
\n	New Line Character	To represent the new-line-character 0x0A. (This corresponds to the character code according to the ASCII table.)

\r	Carriage Return Character	To represent the carriage-return-character 0x0D. (This corresponds to the character code according to the ASCII table.)
\t	Tab Character	To represent the tab-character 0x09. (This corresponds to the character code according to the ASCII table.)

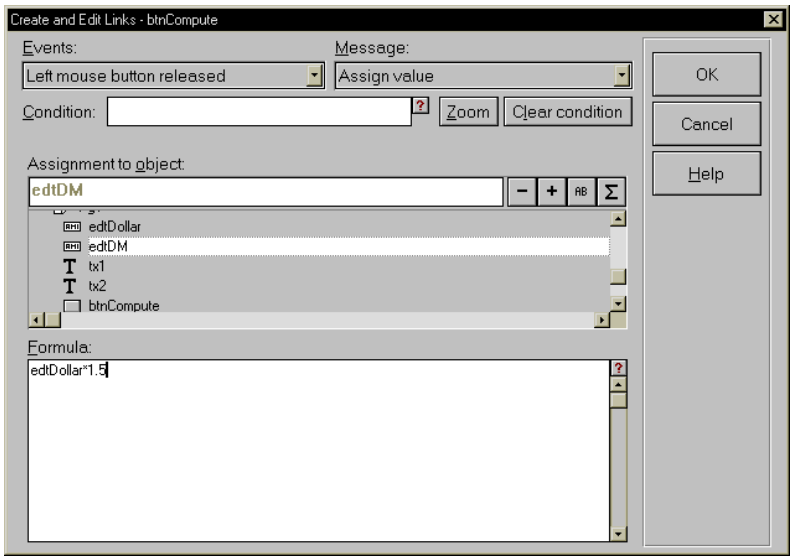
Formulas and functions

mindmap allows for any component to have a value, even if it is not apparent (i.e. command buttons) that this is the case. Some of the components are able to display their value. These include, for example, text and input fields. Other components cannot visualize their value, as is the case for graphical primitives, imported graphics, etc. In either case, it is possible to query the component for its value. A component may also include a formula. The result of the formula is the value of the component.

The values may be of the type numeric, string, or date. Strings are always enclosed in quotation marks. If a string is entered without the quotes, the parser will generate an error message (see 'Parser Error Messages' on page 391). Date constants are also enclosed in quotation marks, but they must follow the national rule of date formatting. Therefore, entering a calendar date depends on the currently selected language DLL (currently either MMDEU.DLL or MMENG.DLL).

Independent of the currently selected language, decimals in numeric constants are always separated by dot (not by comma). This convention makes a mindmap application independent of the language it was built in. However, input fields display numeric values (as well as dates) nationalized.

Assigning a Value via a Link

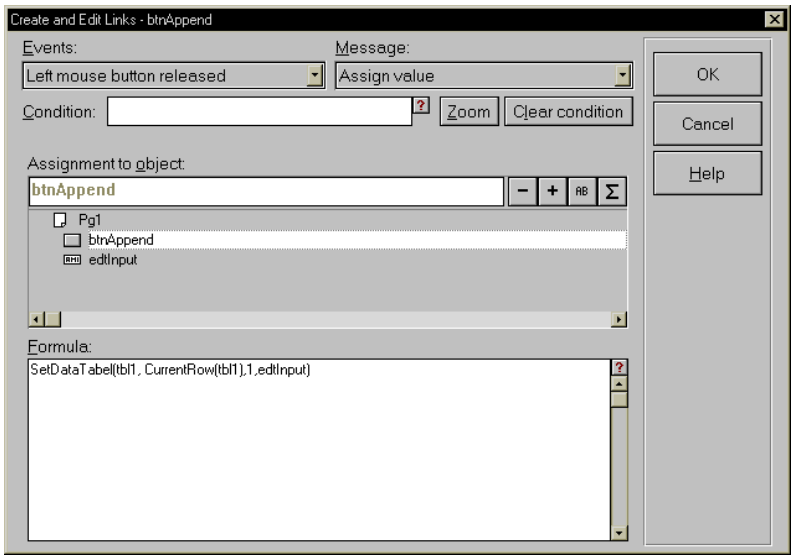


This link dialog box is used to assign a computed value to a component

This link, placed on the OK button, assigns the result of edtDollar multiplied by 1.5 to the input field edtDM.

Functions with an unspecified return value

In contrast to most formulas and functions, mindmap supports various functions which do not return a specific result. These functions are also used by assigning a value. In this case, you assign the value to a component which does not react to value assignments (i.e. command buttons). In such a case, the function is executed without changing the values of components.



mindmap will evaluate the function and ignore the result

In this example, the first column in the selected row in the data table tbl1 receives the value contained in edtInput. The function SetDataTable does not return a value.

➤ More information about assigning values to a component can be found on page 258.

Functions

A function is a built-in routine used to perform a specific calculation. Instead of typing what might be a complex formula, you simply type the function name. mindmap performs the calculations defined by the function.

You include the function name in your own calculation formulas, followed by the values you want the function to use. The information contained in the parentheses are called the parameters (also known as arguments). Most functions have one or more parameters that you must supply in order for mindmap to calculate the results you want.

Almost all mindmap functions contain these three basic parts:

- function name
- a set of parentheses
- the required parameters.

The function name tells mindmap what kind of work to do with the parameters you supply. The parentheses identify where the list of parameters begins and ends. When a function requires more than one parameter, the parameters must be separated with commas.

```
height(Rectangle1)
```

You use functions in your formulas by combining them with component names, operators, values, and other functions. Capitalization is not important when typing functions.

Function parameters may be constants, component names, expressions, or other functions. You may nest one function within another to perform more complex calculations with your data.

mindmap automatically converts arguments passed to functions into the required data type. This applies to either built-in, object-registered or functions imported through MNC files. If a function does not require arguments you should not supply parentheses.

Example:

```
substr(datestr(date), 4,2)
          ; extracts month from date
substr(date,4,2)
          ; the result will be the same because mindmap automatically converts the date to string since this is the required data type for the function substr.
```

General Functions

The following functions are always available. They are either intrinsic to mindmap or have been declared externally in the MMPARSE.MNC.

➤ For further details pertaining to the declaration of functions, please refer to page 479.

abs

Syntax:	abs(<num>)
Description:	Returns the absolute value of a number. If the number is negative, the abs function returns a positive value.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The returned number is of the same data type as the parameter <num>.
Example:	`abs(edt1)` → `10` ; where edt1 contains the number 10 `abs(edt2)` → `10` ; where edt2 contains the number -10
See Also:	sign
Category:	Intrinsic function.

ANSI

Syntax:	ANSI(<text1>, <text2>, <num>)
Description:	Copies the first string into the second one, translating from the IBM-8 character set into the ANSI character set. The numerical value specifies how many characters are to be translated. The len function maybe used to calculate the number of characters available in the source string.
Parameter(s):	<text> is a mindmap component that can receive a text string. The <num > parameter is an integer.

Return Value(s):	A string is returned.
Example:	This function is especially used to translate language dependent special characters like German umlaut characters to the windows character set. `ANSI(edt1,edt2,5)` → `"abcde"` ; where edt2 contains "abcdefg"
See Also:	len
Category:	Declared function.

AppName

Syntax:	AppName (<component name>)
Description:	This function returns a string containing the full path name of the application that contains the given component.
Parameter(s):	name of a component
Return Value(s):	A string
Example:	`AppName (StartButton1)` → `C:\MINDMAP\APPS\LOGON.MM`
Category:	MMLIB

arccos

Syntax:	arccos(<num>)
Description:	Returns the arccosine of a number.
Parameter(s):	The <num> parameter is any expression that yields a number in the range -1 to 1. This is a value in radians.

Return Value(s):	The returned value is in radians. The formula for converting degrees to radians is radians=degrees*(/ 180)
Example:	`arccos(edt1) → 1.2238 ; where edt1 contains .34`
See Also:	arctan, arcsin, sin, cos, tan
Category:	Intrinsic function.

arccosh

Syntax:	arccosh(<num>)
Description:	Returns the hyperbolic arccosine of an angle.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The return value is numeric.
See Also:	sinh, cosh, tanh, arcsinh, arctanh
Category:	Intrinsic function.

arcsin

Syntax:	arcsin(<num>)
Description:	Returns the arcsine of a number.
Parameter(s):	The <num> parameter is any expression that yields a number in the range -1 to 1.
Return Value(s):	The returned value is in radians. The formula for converting degrees to radians is radians=degrees*(/ 180)
Example:	`arcsin(edt1) → .346916 ; where edt1 contains .34`

See Also:	arctan, arccos, sin, cos, tan
Category:	Intrinsic function.

arcsinh

Syntax:	arcsinh(<num>)
Description:	Returns the hyperbolic arcsine of an angle.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The return value is numeric.
See Also:	sinh, cosh, tanh, arccosh, arctanh
Category:	Intrinsic function.

arctan

Syntax:	arctan(<num>)
Description:	Returns the arctangent of a number.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The returned value is in radians. The formula for converting degrees to radians is radians=degrees*(/ 180)
Example:	`arctan(edt1) → 1.5374753` ; where edt1 contains 30.
See Also:	arcsin, arccos, sin, cos, tan
Category:	Intrinsic Function.

arctanh

Syntax:	arctanh(<num>)
Description:	Returns the hyperbolic arctangens of an angle.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The return value is numeric.
See Also:	sinh, cosh, tanh, arcsinh, arccosh
Category:	Intrinsic function.

calc

Syntax:	calc(<text>)
Description:	This function evaluates the statement contained in the text string that is supplied as parameter.
Parameter(s):	The <text> parameter is a string representing a valid parser statement.
Return Value(s):	The outcome of this function depends on the statement. It has the same type that the statement evaluates to.
Example:	If the input field edt1 contains the value of 5 and the input field edt2 contains the text "edt1", then the function `calc("7 * " + edt2)` ; will return 35.
Category:	Intrinsic function.

color

Syntax: color (<num>,<num>,<num>)

Description: This function converts three color values into a 32-bit RGB color value. The return value is calculated using the formula

R*256*256+G*256+B

The lower 8 bits represent the color value for blue, the next 8 bits represent the color value for green and the next 8 bits represent the color value for blue.

```
32..25 24..17 16..9     8..1
unused red    green     blue
```

Parameter(s): The three parameters specify the three color values for R, G and B respectively.

Return Value(s): 32-bit color value.

Example:
```
color (255, 255, 255) → 0x00ffffff
```
(hexadecimal for white)

See Also:

Category: MMLIB

CopyFile

Syntax: CopyFile(<text>, <text>)

Description: Copies the second file onto the first file. If the first file already exists, it will be overwritten.

Parameter(s):	The <text> parameter contains a valid file name with the path if the file is not in the current directory..
Return Value(s):	0 ; OK 1 ; Out of memory 2 ; Source file not found 3 ; Cannot create target file 4 ; Disc full 5 ; Wildcard file copy not successful

Example:	`CopyFile(edt1, edt2)`
	; where edt1 contains the text string "C:\FILE1.TXT" and edt2 contains the text string "FILE2.TXT"
	or
	`CopyFile("c:\\mm", "c:\\mindmap\\*.*")`

See Also:	DeleteFile
Category:	Declared function.

COS

Syntax:	cos(<num>)
Description:	Returns the cosine of a number.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The returned value is in radians. The formula for converting degrees to radians is radians=degrees*(/ 180)
Example:	`cos(edt1)` → `.154251` ; where edt1 contains 30
See Also:	arccos, arcsin, arctan, tan, sin
Category:	Intrinsic function.

cosh

Syntax:	cosh(<num>)
Description:	Returns the hyperbolic cosine of an angle.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	The return value is numeric.
See Also:	sinh, tanh, arcsinh, arccosh, arctanh
Category:	Intrinsic function.

crlf

Syntax:	crlf
Description:	Inserts a carriage return/line feed.
Parameter(s):	none
Return Value(s):	Returns a string containing the carriage return and line feed characters.
Example:	An assign value command to an input field supplies the following results:

```
"Joe "+"Jones"
                    ; Joe Jones
"Joe "+crlf+"Jones"
                    ; Joe Jones
```

Category:	Intrinsic function

date

Syntax:	date

Description:	Supplies the current system date.
Parameter(s):	none.
Return Value(s):	A date in the format set in the MINDMAP.INI file which is dependent on what language (MMDEU.DLL or MMENG.DLL) is selected. Please note that you may set the date format in the application, by using the mask attribute. Then, the format is independent of the MINDMAP.INI.
Example:	`date` → `12.07.1995` (with MMDEU.DLL installed) `date` → `07/12/95` (with MMENG.DLL installed)
See Also:	datestr, strdate, day, year, month, time, sweekday, smonth
Category:	Intrinsic function.

datestr

Syntax:	datestr(<date>)
Description:	Converts <date> into a string. In most cases this conversion is not necessary, since mindmap always attempts to perform the necessary conversions, if there are inconsistencies in the data types. Note that mindmap uses a language specific notation for the resulting string, depending on the currently selected language DLL.
Parameter(s):	The <date> parameter contains a valid date.
Return Value(s):	A string containing the date, as a string in the format defined in the

MINDMAP.INI file or with the mask attribute of the component.

Example: `datestr(edt1)` → `"12.07.1995"`
; where edt1 contains 12.07.1995

Please note again that, if you forget the datestr function in this example, mindmap would convert date to string automatically, if a string is expected instead of a date.

See Also: date, strdate, day, year, month, smonth, time, weekday sweekday

Category: Intrinsic function.

day

Syntax: day(<date>)

Description: Returns the numeric value of the day of the month of the calendar date supplied.

Parameter(s): A string containing the date.

Return Value(s): A string containing a two-digit number.

Example: `day(edt1)` → `12`
; where edt1 contains the current date 12.07.1995 (if MMDEU.DLL is installed)

`day(edt2)` → `12`
; where edt2 contains the current date 07/ 12/ 1995 (if MMENG.DLL is installed)

See Also: month, smonth, year, date, strdate, datestr, time, weekday, sweekday

Category: Intrinsic function

DeleteFile

Syntax:	DeleteFile(<text>)
Description:	The function deletes the file, whose file name is contained in <text>.
Parameter(s):	The <text> parameter contains a valid file name or path and file name if the file is not in the current directory.
Return Value(s):	If the function was not successful, one of the following error values will be returned: ▸ File not found ▸ Path not found ▸ Access denied

Example:	This function is especially used to translate language dependent special characters like German umlaut characters to the windows character set.
	`DeleteFile(edt1)`
	`; where edt1 contains the text string "FILE1.TXT" or "C:\\FILE1.TXT".`
	Please note that this function does not support long file names in or Windows NT.
	`DeleteFile(edt1)`
	`; where edt1 contains the text string "FILE1.TXT" or "C:\\FILE1.TXT"`

See Also:	CopyFile
Category:	Declared function.

exp

Syntax:	exp(<num>)
Description:	Returns the exponential value of the given parameter.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	Exponential value
Example:	`exp (3)` → `20.0855369231877`
See Also:	log, ln
Category:	Intrinsic function.

Format

Syntax:	Format (<text>, <num>)
Description:	This function converts a number into a string. You may select from a variety of format specifiers to control the output.
Parameter(s):	The first parameter specifies the format. It has the following form: %[-][#][0][width][.precision]type Each field of the format specification is a single character or number signifying a particular format option. The simplest format specification contains only the percent sign and a type character (for example, %i). The optional fields (in brackets) control other as-

pects of the formatting. Following are the optional and required fields and their meanings:

Field	Meaning
-	Pad the output value with blanks or zeros to the right to fill the field width, aligning the output value to the left. If this field is omitted, the output value is padded to the left, aligning it to the right.
#	Prefix hexadecimal values with 0x (lowercase) or 0X (uppercase).
0	Pad the output value with zeros to fill the field width. If this field is omitted, the output value is padded with blank spaces.
width	Convert the specified minimum number of characters. The width field is a nonnegative integer. The width specification never causes a value to be truncated; if the number of characters in the output value is greater than the specified width, or if the width field is not present, all characters of the value are printed, subject to the value of the precision field.
precision	Convert the specified

	minimum number of digits. If there are fewer digits in the argument than the specified value, the output value is padded on the left with zeros. The value is not truncated when the number of digits exceeds the specified precision. If the specified precision is zero or omitted entirely, or if the period (.) appears without a number following it, the precision is set to 1.
type	This field may be any of the following character sequences:

Sequence	Meaning
d, i	Insert a signed decimal integer argument (16-bit).
ld, li	Insert a long signed decimal integer argument (32-bit).
u	Insert an unsigned integer argument (16-bit).
lu	Insert a long unsigned integer argument (32-bit).
x, X	Insert an unsigned hexadecimal integer argument in lowercase or uppercase (16-bit).
lx, lX	Insert a long unsigned hexadecimal integer argument in lowercase or uppercase (32-bit).

	The second parameter is the number to be converted.
Return Value(s):	The function returns the converted string.

Example:	If edt1 contains the value 123, the function will convert the following strings
	`Format("%10I",edt1)` → `".......123"`
	`Format("%05I",edt1)` → `"00123"`
	`Format("%-10I",edt1)` → `"123......."`
	`Format("%#x",edt1)` → `"0x7b"`
	`Format("%04x",edt1)` → `"007b"`
	where . represents a space character.

See Also:	FormatString
Category:	Imported function.

FormatString

Syntax:	FormatString(<text>, <text>)
Description:	This function converts a string. You may select from a variety of format specifiers to control the output.
Parameter(s):	The first parameter specifies the format. It has the following form: %[-][width][.precision]s Each field of the format specification is a single character or number signifying a particular format option. The simplest format specification contains only the percent sign and the charac-

ter s (for example, %s). The optional fields (in brackets) control other aspects of the formatting. Following are the optional and required fields and their meanings:

Field	Meaning
.	Pad the output value with blanks to the right to fill the field width, aligning the output value to the left. If this field is omitted, the output value is padded to the left, aligning it to the right.
Width	Convert the specified minimum number of characters. The width field is a nonnegative integer. The width specification never causes a value to be truncated; if the number of characters in the output value is greater than the specified width, or if the width field is not present, all characters of the value are printed, subject to the value of the precision field.
Precision	Convert the specified maximum number of characters.

The second parameter is the string to be converted.

Return Value(s): The function returns the converted string.

Example:	If edt1 contains the string "Hello", the function `Format('%10s',edt1)` will return the string "Hello....." ; where . represents a space character.
See Also:	Format
Category:	Imported function.

frac

Syntax:	frac(<num>)
Description:	This function returns the fractional part of a floating-point value.
Parameter(s):	<num> is a floating-point value
Return Value(s):	A floating-point value

Example:	`frac (3.56)`
	; equals 0.56
	`frac (-5.33)`
	; equals -0.33
	`frac (val(edt1))`
	; equals 0.44 if edt1 contains e. g. "9.44"

See Also:	int, round
Category:	Intrinsic function.

GetFileCount

Syntax:	GetFileCount(<text>)
Description:	Counts the number of files specified

by <text>. Wildcards are permitted.

Parameter(s): The parameter <text> contains a valid MS-DOS directory name.

Return Value(s): An integer is returned.

Example:
```
GetFileCount(edt1) →231
                ; where edt1 contains the
                string C:\WINDOWS\*.*.
```

When specifying paths always use two backslashes instead of single back-slashes.

See Also: CopyFile, DeleteFile

Category: Declared function.

GetHomeDir

Syntax: GetHomeDir(<text>)

Description: This function converts the file name defined by <text> into a full path name representing a file in the application's home directory, provided that <text> is not a full path name itself. The application's home directory is the directory where the application EXE file is.

Parameter(s): <text> Identifies the name of a file and should not contain path and drive specifications.

Return Value(s): <text> The return value is a string representing a full path name based on the name given through the function's parameter.

Example:
```
ReadProfile
("MyApp","UserName","",GetHom
eDir("DEMO.INI"))
```

the above function reads the user's

	name from the INI file in the same directory where the EXE file is (instead of searching the Windows directory).
See Also:	ReadProfile, WriteProfile
Category:	Declared function.

GetModuleHandle

Syntax:	GetModuleHandle(<text>)
Description:	This function retrieves the handle of the specified module.
Parameter(s):	<text> address of name of module
Return Value(s):	Handle of the module, if the function is successful. Otherwise, it is NULL.
Example:	This function maybe used to verify the existence of a running instance of a program. It will return a non-zero value if the specified module name is already running. For instance GetModuleHandle("Excel") will return a positive integer if Excel has already been launched. Please note that the name of the module is not always identical to the name of the associated EXE file; however, a number of tools are available in the public domain to show a list of all running modules.
Category:	Declared MS-Windows kernel function.

GetParent

Syntax: GetParent(<num>)

Description: Retrieves the handle <num> of the given window's parent window (if any).

The following hierarchy applies to a mindmap Application running as an executable file:

mindmap Application Window
    Object Area Window

while the following hierarchy applies to a mindmap application in the Development Environment:

mindmap Frame Window
    MDIClient class
        MDI Child Window
            Object Area Window

Parameter(s): <num> identifies the window whose parent window handle is to be retrieved.

Return Value(s): <num> is the handle of the parent window, if the function is successful. Otherwise, it is NULL, indicating an error or no parent window.

Example:
```
GetParent(MMWindow))
```

See Also: SetWindowText

Category: Declared MS-Windows kernel function.

GetTickCount

Syntax:	GetTickCount
Description:	This function returns the number of milliseconds that have elapsed since Windows was started.
Parameter(s):	None.
Return Value(s):	The function returns a numeric value (32-bit) which represents the number of milliseconds since Windows was started.
Example:	`GetTickCount → 22133234`
Category:	Intrinsic function.

GetWindowText

Syntax:	GetWindowText(<num1>, <text>, <num2>)
Description:	This function copies text of the given window's title bar (if it has one) into a buffer. If the given window is a control, the text within the control is copied.
Parameter(s):	<num1> handle of window <text> address of buffer for text <num2> An integer; maximum number of bytes to copy
Return Value(s):	The length, in bytes, of the copied string, not including the terminating null character. It is zero if the window has no title bar, the title bar is empty, or the <num1> parameter is invalid.
	`GetWindowText(GetParent(MMWin`

```
dow),edt1,256)
```

Since mindmap guarantees the size of a call-by-name argument to be 4096, the third parameter can be any suitable value less than 4096. Under normal circumstances, 256 is sufficient.

See Also: SetWindowText

Category: Declared MS-Windows kernel function.

gsum

Syntax: gsum(<component name>)

Description: The function computes the sum of all components (that have a value associated with them) placed on top of <component name>. It performs a graphical summation. Instead of expecting values as arguments, this function works on components dragged on top of another component.

Parameter(s): A component name.

Return Value(s): A numeric value

Example:
```
gsum(rc1) → 100.2
; where two components are placed on top of rc1, each having a value of 50.1.
```

Category: Intrinsic function.

height

Syntax: height(<component name>)

Description:	It measures in pixels the height of the specified component.
Parameter(s):	The <component name> parameter is a valid component name.
Return Value(s):	An integer
Example:	`height(rc1) → 30` ; where the rectangle rc1 is 30 pixels high
See Also:	width
Category:	Intrinsic function

Hex

Syntax:	Hex (<num>)
Description:	This function converts a numerical value into hexadecimal notation.
Parameter(s):	numerical value (only the integer part is used).
Return Value(s):	string containing the converted hexa-decimal value.
Example:	`hex(1024) → 400` ; returns the string "400"
Category:	MMPSTOOL

hwnd

Syntax:	hwnd(<component name>)
Description:	This function returns the window handle of the window to which the specified component belongs. The following hierarchy applies to a

mindmap Application running as an executable file:

mindmap Application Window
    Object Area Window

while the following hierarchy applies to a mindmap application in the Development Environment:

mindmap Frame Window
    MDIClient class
        MDI Child Window
            Object Area Window

Parameter(s):	The <component name> parameter is a valid component name.
Return Value(s):	none
Example:	This function is generally useful in (imported) functions which require the window handle of the window the component is associated with.
See Also:	GetParent, hwndchild
Category:	Intrinsic function.

int

Syntax:	int(<num>)
Description:	This Function returns a floating-point value representing the largest integer that is less than or equal to <num>.
Parameter(s):	The <num> parameter contains any number.
Return Value(s):	Floating-point result; no error return

Example:	`int(edt1) → 10`
	; where edt1 contains 10.5
	`int(edt2) → -10`

```
                    ; where edt2 contains -9.6.
int (2.8)
                    ; is 2.000000
int (-2.8)
                    ; is -3.000000
```

See Also: frac, round

Category: Intrinsic function.

JulianDate

Syntax: JulianDate (<num>,<num>,<num>)

Description: This function computes the Julian Date from three numerical values of year, month and date. This function is useful to perform date calculations (e.g. number of days between dates, calculation of the day of week).

Parameter(s): The first parameter is the year (if this value is less than 100 it is assumed to be the year of 1900+y).

The second parameter is the month.

The third parameter is the day.

Return Value(s): returns the Julian Date.

Example: `JulianDate (44,10,12)` → `2431376`

See Also: YMDFromJulian

Category: MMLIB

len

Syntax: len(<text>)

Description: Obtains the length of the supplied string.

Parameter(s):	The <text> parameter is a string.
Return Value(s):	Returns an integer.

Example:	`len(edt1) → 7`
	; where edt1 contains the string “abcdefg”.
	If edt1 contains a numeric value instead of a string, the correct expression is the following:
	`len(str(edt1))`

See Also:	str, substr, strpos, strrepl, upper, lower
Category:	Intrinsic function

ln

Syntax:	ln(<num>)
Description:	Determines the natural logarithm of <num>.
Parameter(s):	The parameter <num> is a floating point value and must be greater than zero.
Return Value(s):	A floating point value as the result or 0 as an error value. Please note that the result of ln(1) is also 0! Also note, that if an error occurs the error log reports this function as log10.

Example:	`ln(1) → 0`
	`ln(100) → 4.605170`
	`ln(0.8) → -0.223144`
	`ln(edt1)`
	; edt1 contains a numeri-

cal value

```
ln(val(edt1))
        ; edt1 contains a string of
        numbers
```

See Also: log

Category: Intrinsic function.

log

Syntax: log(<num>)

Description: Returns the common logarithm (base 10) of a number.

Parameter(s): The <num> parameter may have any positive floating point value.

Return Value(s): A floating point value as the result or 0 as an error value.

Please note that the result of log(1) is also 0!

Also note, that if an error occurs the error log reports this function as log10.

Example:

```
log(11) → 1.041392
log(100) → 2
log(0.8) → -0.096910
log(edt1)
        ; edt1 contains a numeri-
        cal value
log(val(edt1))
        ; edt1 contains a string of
        numbers
```

See Also: ln

Category: Intrinsic function.

lower

Syntax:	lower(<text>)
Description:	This function converts all letters in <text> to lowercase.
Parameter(s):	The <text> parameter contains characters.
Return Value(s):	A string containing only lowercase letters.
Example:	`lower(edt1)` → `"abcde"` ; where edt1 contains "ABcDE"
See Also:	upper, str, val
Category:	Intrinsic function.

lstrspn

Syntax:	lstrspn (<text1>,<text2>)
Description:	This function returns the index of the first character in string1 not belonging to string2. This function is useful to remove trailing spaces (or other characters) from strings. Please note that the returned index is zero-based since this is an imported kernel function.
Parameter(s):	Parameter: string to be searched. Parameter: characters to be ignored.
Return Value(s):	returns a zero-based index into string1.
Example:	The statement `substr(edt1,`

```
lstrspn(edt1,' ')+1)
```

will eliminate all trailing spaces from the string in the input field edt1.

Category: MMLIB

MakeDir

Syntax: MakeDir (<text>)

Description: This function creates the directory specified by the string given in the parameter.

Parameter(s): Name of the directory to be created. This name should not contain a terminating backslash character. The name may contain a drive specifier.

Return Value(s): This function returns one of the following numerical values:

0	directory already exists
1	directory has been created
2	cannot create directory (disk may be write protected or specified directory is invalid).

Example: MakeDir("c:\\example")

MMWindow

Syntax: MMWindow

Description: Retrieves the windows handle of the currently active mindmap application window.

Parameter(s): none

Return Value(s): An integer value representing the window handle.

Example: The following operation sets the caption text of the currently active mindmap window.

```
SetWindowText(GetParent(MMWindow),"")
```

Note that, since mindmap application windows are always child windows, their parent is either an MDI child or a tiled window.

See Also: GetParent, SetWindowText

Category: Declared mindmap function.

month

Syntax: month(<date>)

Description: Extracts the numeric value of the month from the date supplied.

Parameter(s): A string containing the date.

Return Value(s): A string containing a two-digit number

Example:

```
month(date)
```

```
month("12.07.1995") → 7
```
(with MMDEU.DLL installed)

```
month("07/12/1995") → 7
```
(with MMENG.DLL installed)

It is important to choose the date format that corresponds to the mindmap language selection.

See Also: day, week, year, date, strdate, datestr, smonth, sweekday

Category: Intrinsic function.

ObjectCount

Syntax:	ObjectCount
Description:	It counts the number of components on the currently visible mindmap page.
Parameter(s):	none
Return Value(s):	An integer value.
Example:	`ObjectCount → 12` ; where the page contains 12 components.
See Also:	ObjectValue
Category:	Intrinsic function.

PageCount

Syntax:	PageCount
Description:	It returns the total number of pages in the currently active MM file.
Parameter(s):	none.
Return Value(s):	An integer value.
Example:	`PageCount → 72` ; where the total number of pages in the application is 72.
See Also:	PageNum, ReportPage
Category:	Intrinsic function

PageNum

Syntax:	PageNum
Description:	It returns the page number of the current page of either a mindmap application, a printer template or an output page object. The function returns a value of 1 for the first page.
Parameter(s):	none.
Return Value(s):	An integer value.
Example:	`PageNum → 6` ; where the function is placed on the 6th page of the application
See Also:	PageCount, ReportPage
Category:	Intrinsic function.

pi

Syntax:	pi
Description:	It computes the value of pi, which is the ratio of the circumference to the diameter of a circle.
Parameter(s):	none.
Return Value(s):	3.14159
Example:	`pi*2 → 6.282`
Category:	Intrinsic function.

PointInObject

Syntax:	PointInObject(<component name>, <x-coordinate>, <y-coordinate>)
Description:	This function determines if a point defined by its x- and y-coordinate lies within a given object.
Parameter(s):	The <component name> parameter is a valid component name. The <x-coordinate> is a horizontal coordinate. The <y-coordinate> is a vertical coordinate.
Return Value(s):	This function returns 1 if the point lies within the given component, otherwise 0.
Example:	Use this function together with the functions GetMouseX and GetMouseY to determine if (and where) the mouse is positioned with respect to a given component.
See Also:	GetMouseX, GetMouseY
Category:	Intrinsic function.

rand

Syntax:	rand
Description:	This function returns a pseudo random value between 0 and 32767.
Parameter(s):	None
Return Value(s):	The value returned is a pseudo random number.

Example: Use the following conversions if applicable

```
r = rand/32768 → 0 <= r < 1
r = 10*rand/32768 → 0 <= r < 10
```

Category: Intrinsic function.

ReadProfile

Syntax: ReadProfile(<text1>, <text2>, <text3>, <text4>)

Description: This function returns a string associated with a given entry in an INI file. The INI file is assumed to be in the windows directory if a full path name is not supplied.

Parameter(s):

text1	section name
text2	key name
text3	default value to be returned, if the key does not exist.
text4	name of the INI file.

Return Value(s): A string representing the entry in the INI file or the given default value if the section and keyword could not be found.

Example:

```
ReadProfile
("windows","device","No
printer installed","WIN.INI")
```

returns the currently installed default printer or the message No printer installed if such an entry could not be found.

If the file name is specified without path, it is assumed to be in the Windows directory. Use the GetHomeFile function to specify an .INI file in the

	mindmap home directory.
See Also:	WriteProfile
Category:	Declared function

round

Syntax:	round(<num1>, <num2>)
Description:	Rounds the floating point value <num1>. <num2> is the number of digits to the right of the decimal point.
Parameter(s):	<num1> A floating point value <num2> An integer, the number of decimal digits; optional
Return Value(s):	A floating point value.

Example:	`round(PI,3)` → `3.142`
	`round(PI)` → `3`
	; PI equals 3.14159
	`"$ "+str(round(edt1,2))` → `"$ 132.45"`
	; the contents of edt1 is 132.448123

See Also:	frac, int
Category:	Intrinsic function

SetWindowText

Syntax:	SetWindowText(<num>, <text>)
Description:	This function sets the title (caption bar) of the currently active window to be <text>
Parameter(s):	<num> is the window handle

<text> is a string

Return Value(s): This Function does not return a value.

Example: The function

```
SetWindowText(GetParent(MMWin
dow),"Print")
```

changes the text in the caption bar of the mindmap Application Window to contain the word Print.

See Also: GetWindowText, GetParent

Category: Declared MS-Windows function.

ShowWindow

Syntax: ShowWindow(<num1>, <num2>)

Description: The ShowWindow function sets the given window's visibility state.

Parameter(s): <num1> is the window's handle

<num2> is an integer; the window visibility flag, defined as follows

0	hide the window
1	show the window
2	minimize the window (show as icon)
3	maximize the window
9	restore the window to its original size and position

Return Value(s): Returns equal zero, if the window was previously hidden;

returns nonzero, unequal 0; if the window was previously visible.

Example:	The operation: `ShowWindow(GetParent(MMWindow),2)` minimizes the currently active mind-map application.
See Also:	GetParent, MMWindow
Category:	Declared MS-Windows function.

sign

Syntax:	sign(<num>)
Description:	Determines if <num> is positive or negative.
Parameter(s):	<num> A floating point value
Return Value(s):	A The function returns -1 if the given parameter is less than zero, 1 if the parameter is greater than zero and 0 if the given value is 0.floating point value.

Example:	`sign(-1.23)` → `-1`
	`sign(1.23)` → `1`
	`sign(0)` → `0`

See Also:	abs
Category:	Intrinsic function

sin

Syntax:	sin(<num>)
Description:	Returns the sine of an angle that is measured in radians. The formula for converting degrees to radians is radi-

ans=degrees*(/ 180).

Parameter(s): The <num> parameter is any expression that yields a number.

Return Value(s): A numeric value.

Example: `sin(edt1)` → `.49999999` ; where edt1 contains 30*(pi/ 180)

See Also: tan, cos, arcsin, arctan, arccos

Category: Intrinsic function.

sinh

Syntax: sinh(<num>)

Description: Returns the hyperbolic sine of an angle.

Parameter(s): The <num> parameter is any expression that yields a number.

Return Value(s): A numeric value.

See Also: cosh, tanh, arcsinh, arccosh, arctanh

Category: Intrinsic function.

smonth

Syntax: smonth(<date>)

Description: Returns the name of the month of <date> using the installed language library.

Parameter(s): A date

Return Value(s): A string

Example:	`smonth(date)` → `"August"`
	; date "08/ 09/ 95" (English Format)
	`smonth(edt1)`
	; edt1 contains a date.

See Also:	day, ,week, month, year, date, strdate, datestr, sweekday
Category:	Intrinsic function

sqrt

Syntax:	sqrt(<num>)
Description:	Returns the square root of <num>.
Parameter(s):	The <num> parameter is any expression that yields a non-negative number.
Return Value(s):	A floating point number. Note that this function returns 0 for a square root of a negative number. However, an entry in the parser error window is generated.
Example:	`sqrt(edt1)` → `12` ; where edt1 contains 144.
Category:	Intrinsic function.

str

Syntax:	str(<num>)
Description:	Translates a number <num> into a string.
Parameter(s):	The <num> parameter contains a number value.
Return	A string representation of <num>.

Value(s):

Example:
```
str(edt1) → "123"
            ; where edt1 contains the
            integer 123.
Str(123) -→ "123"
str(-123.45) →"-123.45"
```

See Also: val, strdate, datestr, strpos, substr, strrepl

Category: Intrinsic function

strdate

Syntax: strdate(<text>)

Description: Converts <text> into a valid date if possible. You may use this function to force a conversion where other functions require an argument of the type date.

Keep in mind that mindmap will attempt to automatically convert strings to date values where dates are required, even if a string is supplied.

Parameter(s): The parameter <text> contains date.

Return Value(s): A valid date.

Example:
```
strdate(edt1) →
            12.07/12/.19975
            ; where edt1 contains
            "12.07/ 12/ .19975".
```

See Also: date, datestr, str, substr, strpos, strrepl

Category: Intrinsic function.

strpos

Syntax:	strpos(<text1>, <text2>)
Description:	Searches for the first occurrence of <text2> in <text1>.
Parameter(s):	<text1> and <text2> contain strings.
Return Value(s):	An integer value with the beginning position (1-based). The value 0 means that nothing was found.

Example:	`strpos(edt1, edt2)` → `2` ; where edt1 contains "abcdef" and edt2 contains "bc".
	`Strpos(edt1,"file")`
	`strpos("c:\\files\\*.*",edt2)`

See Also:	str, substr, upper, lower, strdate, datestr, strrepl
Category:	Intrinsic function.

strrepl

Syntax:	strrepl(<text1>, <text2>, <text3>)
Description:	Replaces all occurrences of <text2> with <text3> in <text1>.
Parameter(s):	All parameters contain strings
Return Value(s):	An integer; the number of replacements.
Example:	`strrepl(edt1, edt2, edt3)` ; where edt1 contains "John Miller", edt2 contains "John", and edt3 contains "Pete". After the

replacement, edt1 contains "Pete Miller" and the return value is 1 (for 1 replacement).

See Also: str, strpos, substr, upper, lower, strdate, datestr

Category: Intrinsic function.

substr or substring

Syntax: substr(<text>, <num1>, [<num2>])
substring(<text>, <num1>, [<num2>])

Description: Extracts from <text> a substring starting at position <num1> and with a length of <num2> (1-based).

Parameter(s): <text> contains a string, <num1> designates the starting position, <num2> the desired length of the substring to be extracted. The parameter <num2> is optional; without this value the returned string is from the position to the end of <text>.

Return Value(s): A string

Example:

```
substr(edt1,2,2) → "bc"

substr(edt1,2) → "bcdef"
        ; where edt1 contains "abcdef".

substr(edt2,1,2) + " " + date
        → "Fr 08/04/95"
        ; where edt2 contains "Friday" and date is "08/ 04/ 95"

substr(sweekday(date),1,2) →
```

```
                                    "Fr"
                                    ; where date is
                                    "08/ 04/ 95"
```

See Also: str, strpos, upper, lower, strdate, datestr, strrepl

Category: Intrinsic function.

sweekday

Syntax: sweekday(<date>)

Description: Returns the name of the weekday of <date> using the installed language library.

Parameter(s): A date

Return Value(s): A string

Example:

```
sweekday(date) → "Wednesday"
                ; date "08/ 09/ 95"
                (English Format)
sweekday(edt1)
                ; edt1 contains a date
```

See Also: day, ,week, month, year, date, strdate, datestr, sweekday

Category: Intrinsic function

tan

Syntax: tan(<num>)

Description: Returns the tangent of an angle measured in radians. The formula for converting degrees to radians is radians=degrees * (/ 180).

Parameter(s): The <num> parameter is any expres-

	sion that yields a number.
Return Value(s):	A floating point.
Example:	`tan(edt1)` → `0.57735` ; where edt1 contains 30*(pi/ 180)
See Also:	sin, cos, arctan, arcsin, arccos
Category:	Intrinsic function.

tanh

Syntax:	tanh(<num>)
Description:	Returns the hyperbolic tangent of an angle.
Parameter(s):	The <num> parameter is any expression that yields a number.
Return Value(s):	A numeric value.
See Also:	sinh, cosh, arcsinh, arccosh, arctanh
Category:	Intrinsic function.

time

Syntax:	time
Description:	Supplies the current system time in 24h notation.
Parameter(s):	none
Return Value(s):	A time.
Example:	time → 12:43:12
See Also:	date

Category:	Intrinsic function.

trim

Syntax:	trim (<text>)
Description:	This function removes trailing spaces from the given string.
Parameter(s):	String to be transformed.
Return Value(s):	This function returns the transformed string.
Example:	`trim('abcdef ') → 'abcdef'.`
Category:	MMLIB

upper

Syntax:	upper(<text>)
Description:	Returns a string consisting of the uppercase equivalent of the <text> parameter. CharCharacters that lack an uppercase equivalent in the ANSI character set are returned unchanged.
Parameter(s):	The <text> parameter contains a string.
Return Value(s):	A string.
Example:	`upper(edt1) → "ABC/+:?="` ; where edt1 contains "abc/ +:?=".
See Also:	lower, val, str, strpos, substr, strrepl
Category:	Intrinsic

val

Syntax:	val(<text>)
Description:	Returns the numerical equivalent of the supplied <text>, for use with formulas involving numbers or numeric functions.
Parameter(s):	The <text> parameter contains a string representing a number
Return Value(s):	A numerical value.

Example:	`val(edt1)` → `123`
	;where edt1 contains the string "123".
	`Val(substr(edt2,49,.2))` → `45`
	; where edt2 contains the string "1234567890"

See Also:	str, int, frac
Category:	Intrinsic function.

weekday

Syntax:	weekday (<date>)
Description:	This function determines the day-of-week from a given date.
Parameter(s):	The function expects a date value. Use the strdate function to convert a string into a date value.
Return Value(s):	1 = Monday 2 = Tuesday 3 = Wednesday 4 = Thursday 5 = Friday

	6 = Saturday 7 = Sunday
Example:	The function weekday(date) will return the weekday index for today. The function `weekday(strdate("12/25/97"))` will return the weekday index for Christmas 1997.
See Also:	sweekday
Category:	MMLIB

width

Syntax:	width(<component name>)
Description:	It measures in pixels, the width of the specified component.
Parameter(s):	The <component name> parameter is a valid component name.
Return Value(s):	An integer.
Example:	`width(rc1)` → `50` ; where the rectangle rc1 is 50 pixels wide.
See Also:	height
Category:	Intrinsic function.

WinHelp

Syntax:	WinHelp (<num>,<text>,<num>,<text>)

Description: This function directly invokes the windows help system.

Parameter(s): The first parameter is the window handle of the parent window calling the help system. Please always use the result of the function MMWindow.

The second parameter is the name of the help file.
The third parameter is one of the following constants:
2 = Close the Windows help system for the specified help file.
3 = Display the contents page of the specified help file.
261 = Show a help screen for the keyword specified in parameter 4. Please note that this keyword must be defined in the help file for this function to be successful.

The fourth parameter is the keyword if the third parameter is 261. Otherwise this parameter should be an empty string.

Return Value(s): The return value is non-zero if the function was successful. Otherwise zero.

Example:
```
WinHelp(mmwindow,'EXAMPLE.HLP',261,'Options')
```

Category: MMPARSE.MNC

WriteProfile

Syntax: WriteProfile(<text1>, <text2>, <text3>, <text4>)

Description: This function places a string associated with a given entry into an INI file. The INI file is assumed to be in

	the windows directory if a full path name is not supplied.	
Parameter(s):	text1	section name
	text2	key name
	text3	desired value for the given key
	text4	name of the INI file.
Return Value(s):	none	

Example:	The function:
	`WriteProfile ("MyApp","UserName",edt1 ,"MYAPP.INI")`
	writes the contents of the input field with the name edt1 to the file MYAPP.INI in the windows directory. Assuming that edt1 contains the name of the user, his name may later be retrieved by the operation:
	`ReadProfile ("MyApp","UserName","","MYAPP .INI")`
	which is normally used in a value assignment to the input field edt1, again.	

See Also:	ReadProfile	
Category:	Declared function	

xpos

Syntax:	xpos(<component name>)
Description:	It returns the upper left position of the specified <component name> measured in pixels from the top of the screen.

Parameter(s):	The <component> contains a valid component name.
Return Value(s):	An integer.
Example:	`xpos(edt1)` → `56` ; where edt1 is positioned at the 56th pixel from the top of the screen.
See Also:	ypos
Category:	Intrinsic function.

year

Syntax:	year(<date>)
Description:	Returns the year of the calendar date supplied.
Parameter(s):	A string containing the date.
Return Value(s):	A string containing a four-digit number.

Example:	`year(date)` → `"1995"`
	`year(edt1)` → `"1995"`
	; where edt1 contains the current date 12.07.1995 or 12.07.95

See Also:	day, month, date, strdate, datestr, sweekday, smonth
Category:	Intrinsic function

YMDFromJulian

Syntax:	YMDFromJulian (<num>)
Description:	This function converts a Julian Date into a mindmap date value. This func-

	tion is useful to reconvert the result of date calculations (e.g. number of days between dates, calculation of the day of week).
Parameter(s):	The parameter is a numeric value representing a Julian date.
Return Value(s):	The return value is a mindmap date. Use the datestr function to convert this date into a string.
Example:	`datestr(YMDFromJulian(JulianDate(44,10,12)+14))` → `24.10.1944`
See Also:	JulianDate, datestr
Category:	MMLIB

ypos

Syntax:	ypos(<component name>)
Description:	It returns the upper left position of the specified <component name> measured in pixels from the left of the screen..
Parameter(s):	The <component name> contains a valid component name.
Return Value(s):	An integer.
Example:	`ypos(edt1)` → `152` ; where edt1 is positioned at the 152nd pixel from the left of the screen.
See Also:	xpos
Category:	Intrinsic function

Component Specific Functions

Some components register their own functions. These are always available, as long as the corresponding component library (*.MDL) has been loaded by mindmap.

Database Functions

The database functions are declared by MMBASE.MDL, that is part of the mindmap standard installation. If this file is loaded as specified in the MINDMAP.INI, the following functions are available:

dbBaseName

Syntax:	dbBaseName(<database>)
Description:	This function determines the name of the database, or in case of ODBC, the name of the data source to which <database> belongs.
Parameter(s):	The <database> parameter corresponds to a database name, as defined in mindmap.
Return Value(s):	A string.
Example:	`dbBaseName(db1)`
See Also:	dbTableName, dbFieldName
Category:	Declared by MMBASE.MDL

dbCurrentRow

Syntax:	dbCurrentRow(<database>)
Description:	This function calculates the current record number in the current result set.. The first record number is 1.
Parameter(s):	The <database> parameter corresponds to a database name, as defined in mindmap.
Return Value(s):	An integer.

Example:	`dbCurrentRow(db1)`
	`dbCurrentRow(db1)+"/"`
	`+dbRowCount(db1)`
	;this expression displays the current position in a result set, e. g. 11/ 131 would mean, the 11th record of 131 records

See Also:	dbRowCount
Category:	Declared by MMBASE.MDL

dbFieldCount

Syntax:	dbFieldCount(<database>)
Description:	Determines the number of columns of the database table defined by the database object supplied as parameter.
Parameter(s):	The <database> parameter corresponds to a database name, as defined in mindmap.
Return Value(s):	An integer.

Example: `dbFieldCount(db1) → 13`
; if the table db1 consists of 13 columns

Category: Declared by MMBASE.MDL

dbFieldName

Syntax: dbFieldName(<database>,<num>)

Description: This function determines the name of the column in the database table <database> at position <num> (1-based).

Parameter(s): The <database> parameter corresponds to a database name, as defined in mindmap.

Return Value(s): A string.

Example: `dbFieldName(db1,2) →`
`"FirstName"`
if the second field in the database object is named FirstName.

See Also: dbBaseName, dbTableName

Category: Declared by MMBASE.MDL

dbGetDate

Syntax: dbGetDate(<database>,<text>)

Description: This function converts a date into the correct format depending on the database used. Use this function in database queries issued through SQL Exec

commands, e.g. that are not generated automatically through the database command Search for Fields.

Parameter(s): <database> name of the database component

<text> name of the component containing the date that has to be converted

Return Value(s): A string

Example:
```
dbGetDate(db1,edt1) → {(d
          '1995-08-08'})
          ; for an ODBC data
          source
```

The following statement may be used in a SQL Select command, assuming that edtBirth has a value of "01/01/1965":

```
"where BirthDate > " + dbGet-
Date(dbEmployees,edtBirth)
```

The next statement finds all database records where the column BirthDate lies between two boundaries edtBirthBegin and edtBirthEnd:

```
"where BirthDate between " +
dbGetDate(dbEmployees,edtBirt
hBegin) + " and " + dbGet-
Date(dbEmployees,edtBirthEnd)
+ ")"
```

Category: Declared by MMBASE.MDL

dbIsOpen

Syntax: dbIsOpen(<database>)

Description: This function tests if the database <database> is open, in which case a

	conconnect operation to the database has been
	sucsuccessful and a cursor has been es
	tablished.tablished.
Parameter(s):	The <database> parameter corresponds to a database name as defined in mindmap.
Return Value(s):	0, if database is not open
	1, if database is open
Example:	`dbIsOpen(db1)` → `0` ; meaning that the database db1 is not currently opened.
Category:	Declared by MMBASE.MDL

dbRowCount

Syntax:	dbRowCount(<database>)
Description:	This function supplies the number of records in the result set.
	Please not that it depends on the database driver how this function works. In general, most ODBC drivers are not capable of returning the number of rows in the result set, unless the last record in the result set has been fetched. This implies that, to be safe, a “go to last record” command should be executed before using this function. The function returns -1 if the number of records in the database is unknown.
Parameter(s):	The <database> parameter corresponds to a database name, as defined

	in mindmap.
Return Value(s):	An integer.
Example:	`dbRowCount(db1)` `dbCurrentRow(db1)+"/"+ dbRowCount(db1)` ;this expression displays the current position in a result set, e. g. 11/ 131 would mean, the 11th record of 131 records
See Also:	dbCurrentRow
Category:	Declared by MMBASE.MDL

dbSQLSearch

Syntax:	dbSQLSearch(<database>)
Description:	Retrieves the "WHERE"-part of a SQL SELECT statement, according to the components to which the database is connected and for which the "search" option has been set. Please note that the "WHERE"-part is also dependent on the "Search Mode"-settings of an input field,. This is set by setting the attribute symbolized by the magnifier glass on the attribute toolbox of the component.
Parameter(s):	The <database> parameter corresponds to a database name, as defined in mindmap.
Return Value(s):	A string.
Example:	`dbSQLSearch(db1)` → `((ADDNO = 2 and COMPANY = "MGM") and (LASTNAME like`

```
"R%" or LASTNAME like "S%"))
```

In this example there are 3 input fields connected to db1:

edt1 with contents 2 and Search Mode is <f> = <a>

edt2 with contents "MGM" and Search Mode is <f> = <a>

edt3 with contents "R;S" and Search Mode is <f> like "<a>%"

Category: Declared by MMBASE.MDL

dbTableName

Syntax: dbTableName(<database>)

Description: This function determines the name of the table in the database which is represented by the component <database> on the screen.

Parameter(s): The <database> parameter corresponds to a database name, as defined in mindmap.

Return Value(s): A string.

Example: `dbTableName(db1)` → `Address`

See Also: dbBaseName, dbFieldName

Category: Declared by MMBASE.MDL

Input Field Functions

The data table functions are declared by MMEDIT.MDL and/ or MMEDIT.MDL, that is part of the mindmap standard installation. If these files are loaded as specified in the MINDMAP.INI, the following functions are available:

edtGetCol

Syntax:	edtGetCol(<input field >)
Description:	This function determines the column position of the caret in an input field. You may want to use this function to extract a portion of the contents of an input field corresponding to the currently selected text. Please note that this function is valid only if the input field is visible. It is not applicable for input fields that are not on the active page.
Parameter(s):	The <input field> parameter is the name of an input field.
Return Value(s):	This function returns the 1-based column number which identifies the caret position or 0 if the input field is invalid or does not exist.
Example:	`edtGetCol (edt1)` → `4` ; where the cursor has been used to select the 4th column in the input field edt1
See Also:	edtGetRow, edtSetPos
Category:	MMEDIT.MDL

edtGetRow

Syntax:	edtGetRow(<input field >)
Description:	This function determines the row number of the caret in a multiline input field. For a single line input field this function will always return 1. You may want to use this function to extract a portion of the contents of an input field corresponding to the currently selected text. Please note that this function is valid only if the input field is visible. It is not applicable for input fields that are not on the active page.
Parameter(s):	The <input field> parameter is the name of an input field.
Return Value(s):	This function returns the 1-based row number which identifies the caret position or 0 if the input field is invalid or does not exist.
Example:	`edtGetRow (edt1) → 3` ; where the cursor has been put into the 3rd row of the multiline input field edt1
See Also:	edtGetCol, edtSetPos
Category:	MMEDIT.MDL

edtSetPos

Syntax:	edtSetPos(<input field >, <row number>, <column number>)
Description:	This function sets the caret to the po-

	sition specified by the given row and column number. Please note that this function is valid only if the input field is visible. It is not applicable for input fields that are not on the active page.
Parameter(s):	The <input field> parameter is the name of an input field. The <row number> defines the row to which the caret is to be placed. This value must always be 1 for single line input fields. The <column number> defines the column to which the caret is to be placed.
Return Value(s):	This function does not return a value.
Example:	`edtSetPos (edt1, 2, edt2)` → caret set to row 2, column 5 ; where edt1 is a multiline input field and edt2 contains the number 5.
See Also:	edtGetCol, edtGetRow
Category:	MMEDIT.MDL

Data Table Functions

The data table functions are declared by MMDATA.MDL, that is part of the mindmap standard installation. If these files are loaded as specified in the MINDMAP.INI, the following functions are available:

Columns

Syntax:	Columns (<data table name>)
Description:	Determines the number of columns in a data table.
Parameter(s):	Name of a data table
Return Value(s):	Number of columns as an integer value
Example:	`Columns (dt1) → 12` ; where the data table dt1 contains 12 columns
See Also:	Rows
Category:	MMDATA.MDL

CurrentCol

Syntax:	CurrentCol (<Object name>)
Description:	Returns the number (1-based) of the column of a data table which has the input focus.
Parameter(s):	Name of a data table.
Return Value(s):	Number of the selected column as an integer.
Example:	`CurrentCol (dtb1) → 4` ; where the cursor has been used to select the 4th column
Category:	MMDATA.MDL

CurrentRow

Syntax:	CurrentRow(<datatable>)
Description:	Determines the number of the current row (1-based).
Parameter(s):	<datatable> component name
Return Value(s):	An integer

Example:	`CurrentRow(tbl1) → 5`
	; if the 5th row of tbl1 is selected
	`str(CurrentRow(tbl1))+ " / "+ str(Rows(tbl1)) → "5 / 501"`
	; displays the current row in the datatable tbl1, e. g. 5/ 501 would mean, the 5th row of 501 rows.

See Also:	Rows
Category:	Declared by MMDATA.MDL

FirstMarkedRow

Syntax:	FirstMarkedRow(<datatable>)
Description:	Retrieves the number of the first marked row in a multiple selection data table (1-based).
Parameter(s):	<datatable> component name
Return Value(s):	An integer for the row position or a negative number if nothing is selected.
Example:	`FirstMarkedRow(tbl1) → 5`

```
                    ; the 5th, 7th and 8th row
                    of tbl1 are selected
```

See Also: IsRowMarked

Category: Declared by MMDATA.MDL

IsRowMarked

Syntax: IsRowMarked(<datatable>,<row>)

Description: Checks if the row number <row> is highlighted (1-based). Please note that this is only defined for multiple-selection data tables. The function returns a negative value if an error has occurred.

Parameter(s): <datatable> component name

<row> An integer; number of the row

Return Value(s): 0 ; not highlighted

1 ; highlighted

Example:
```
IsRowMarked(tbl1,3)
                    ; checks if the 3rd row of
                    data table tbl1 is high-
                    lighted
```

See Also: FirstMarkedRow

Category: Declared by MMDATA.MDL

Rows

Syntax: Rows(<datatable>)

Description: Determines the number of rows in the Datatable <datatable>

Parameter(s): <datatable> component name

Return Value(s): An integer; number of rows

Example:
```
Rows(tbl1) -→ 124
                ; the Datatable tbl1 consists of 124 rows.
str(CurrentRow(tbl1))+
                ' / '+
                str(Rows(tbl1)) →
                '3 / 124'
                ; displays the current row, 3, in the data table tbl1, which contains 124 rows.
```

See Also: CurrentRow

Category: Declared by MMDATA.MDL

SetDataTable

Syntax: SetDataTable(<datatable>,<row>,<column>,<text>)

Description: Sets the value <text> into the data table at position [<row>,<column>].

Parameter(s): <datatable> component name

<row> An integer

<column> An integer

<text> A string

Return Value(s): none

Example:
```
SetDataTable(tbl1,3,2,'Hello world!')
                ; The string 'Hello world!' is set into the data table tbl1 at position [3,2] which means the 3rd row, 2nd column
```

Category: Declared by MMDATA.MDL

List Box/ Combo Box functions

The list box and combo box functions are declared by MMCOMBO.MDL, that is part of the mindmap standard installation. If this file is loaded as specified by the MINDMAP.INI, the following functions are available:

CursorPos

Syntax:	CursorPos(<list box/ combo box>)
Description:	Determines the number of the current row (1-based).
Parameter(s):	<list box/ combo box> component name
Return Value(s):	An integer

Example:	`CursorPos(lst1) → 3`
	; the 3rd row is selected in list box lst1
	`str(CursorPos(lst1))+" / " +str(LineCount(lst1) ) → "5 / 104"`
	;this expression displays the navigation in the list lst1, e. g. 5/ 104 would mean that the cursor is positioned at the 5th entry of 104 entries

See Also:	LineCount, SelCount
Category:	Declared by MMCOMBO.MDL

LineCount

Syntax:	LineCount(<list box>)

Description:	Determines the number of all entries in the list box.
Parameter(s):	<list box> component name
Return Value(s):	An integer

Example:	`LineCount(lst1) → 382`
	; list box lst1 has 382 entries
	`str(CursorPos(lst1))+" / "`
	`+str(LineCount(lst1)`
	`) → "5 / 104"`
	;this expression displays the navigation in the list box lst1, e. g. 5/ 104 would mean that the cursor is positioned at the 5th entry of 104 entries

See Also:	CursorPos, SelCount
Category:	Declared by MMCOMBO.MDL

SelCount

Syntax:	SelCount(<list box >)
Description:	Determines the number of selected rows in a list box if the list box has either the multiple selection or the extended selection style.
Parameter(s):	<list box/combo box> component name
Return Value(s):	An integer
Example:	`SelCount(lst1) → 3` ; 3 entries are selected
See Also:	CursorPos, LineCount
Category:	Declared by MMCOMBO.MDL

Output page functions

The report functions are declared by MMREPORT.MDL, which is part of the mindmap standard installation. If this file is loaded as specified in the MINDMAP.INI, the following functions are available:

ReportPage

Syntax:	ReportPage(<component name>)
Description:	Retrieves the currently active page number of a report.
Parameter(s):	<component name> name of the report
Return Value(s):	An integer
Example:	`ReportPage(rpt1)` → `4`
See Also:	ReportPageCount, PageNum
Category:	Declared by MMREPORT.MDL

ReportPageCount

Syntax:	ReportPageCount(<component name>)
Description:	Retrieves the number of pages in a report.
Parameter(s):	<component name> name of the report
Return Value(s):	An integer
Example:	`ReportPageCount(rpt1)` → `14`
See Also:	ReportPage, PageCount

Category: Declared by MMREPORT.MDL

VBX Functions

These functions are available if (i) a VBX control is installed and (ii) the mindmap VBX library (MMVBX.MDL) has been loaded.

GetParam

Syntax:	GetParam(<VBX component>, <index>)
Description:	This function is valid only during the response to events generated by a VBX component. It retrieves a parameter which has been provided by the VBX component.
Parameter(s):	The <VBX component> parameter references the VBX component which generated the event. The <index> parameter specifies which of the list of parameters has to be retrieved (1-based).
Return Value(s):	The function returns the appropriate parameter as a string regardless of what type the VBX component has assigned to it.
Example:	This function can be used in cases where a VBX event supplies parameters that a Visual Basic program would normally be able to interpret. If the documentation for a VBX control (placed as vbx1 in mindmap) defines an event through

```
Sub Sample_Change ([Index As
Integer])
```

the function GetParam(vbx1,1) will return the value of Index as a string.

See Also: GetProp, SetProp

Category: MMVBX.MDL

GetProp

Syntax: GetProp (<VBX component name>,<text>)

Description: This function returns the value of a property of the given VBX component. The property is defined by the second parameter.

Parameter(s): The <VBX component> name as it has been declared in the application.

A <text> parameter specifying the name of the property as the VBX has declared it.

Return Value(s): A value corresponding to the property value.

Example: `GetProp (vbx1,"Height")` → `37` ;where the height of the placed VBX instances equals 37 pixels.

See Also: SetProp, GetParam

Category: Declared in MMVBX.MDL

SetProp

Syntax: SetProp (<VBX component name>,<text>,<text>)

Description:	This function directly manipulates the property of a VBX component.
Parameter(s):	The first parameter is the name of the VBX component. The second parameter is the name of the property to be changed. The third parameter is the property value as string. Please convert numeric values to a string even if the specified property requires a numeric value.
Return Value(s):	none.
See Also:	GetProp
Category:	MMVBX.MDL

MCI Functions

These functions are available if (i) an MCI driver is installed for Windows and (ii) the MCI mindmap library (MMVFW.MDL) has been loaded.

mciGetAlias

Syntax:	mciGetAlias (<Multimedia component name>)
Description:	This function returns the currently used alias name for the given multimedia component. This name may be required by certain MCI driver specific functions.
Parameter(s):	Name of a multimedia component.
Return Value(s):	This function returns a string containing the MCI alias name.

Example:	`mciGetAlias(mci1)` → `3061` ; where 3061 represents some (arbitrary) internal identifier, pointing to the component mci1.
See Also:	mciSendString
Category:	MMVFW.MDL

mciGetFileName

Syntax:	mciGetFileName (<Multimedia component name>)
Description:	This function returns the name of the file that has been opened by the given multimedia component.
Parameter(s):	Name of a multimedia component.
Return Value(s):	This function returns a string containing the name of a multimedia file.
Example:	`mciGetFileName(mci1)` → `C:\VFW\MINDMAP.AVI`
Category:	MMVFW.MDL

mciGetLength

Syntax:	mciGetLength (<Multimedia component name>)
Description:	This function returns the length of a multimedia component. The unit of this value depends on the currently selected multimedia device. (A Video for Windows file (*.AVI) returns the length in Frames).
Parameter(s):	Name of a multimedia component.
Return	Numerical value specifying the length

Value(s):	of the file in device specific units.
Example:	The multimedia command "Seek" with a value as of `mciGetLength(mci1) / 2` will play the mci component named mci1 from its middle position.
See Also:	mciGetLength
Category:	MMVFW.MDL

mciGetMediaName

Syntax:	mciGetMediaName (<Multimedia component Object name>)
Description:	This function returns the media name of the currently selected media, if you load a new mci file at run time, i.e. via drag&drop. The value of the return parameter is dependent on the type of media device which is loaded.
Parameter(s):	Name of a multimedia component.
Return Value(s):	Generally a file name, but this depends on the media device type.
Category:	MMVFW.MDL

mciGetMode

Syntax:	mciGetMode (<Multimedia component name>)
Description:	This function returns a string which describes the current state of the multimedia component. Please note that this string depends on the type of MCI driver.
Parameter(s):	Name of a multimedia component.

Return Value(s):	A string describing the state of the component. For a Video for Windows component (*.AVI) this function may return "stopped" "playing"
Example:	`mciGetMode (mci1)` → `"stopped"` ; where the playing of the file has been stopped.
Category:	MMVFW.MDL

mciGetPosition

Syntax:	mciGetPosition (<Multimedia component name>)
Description:	This function returns the current position of a multimedia component. The unit of this value depends on the currently selected multimedia device. (A Video for Windows file (*.AVI) returns the position in Frames).
Parameter(s):	Name of a multimedia component.
Return Value(s):	Numerical value specifying the current position of the file in device specific units.
Example:	`mciGetPosition (mci1)` → `124` ; where the AVI file is currently positioned to frame number 124.
	You may want to use this function to continuously display the position of a playing multimedia device by assigning the result of this function to a text component through a link on the multimedia event "Position Changed".
See Also:	mciGetPositionString

Category:	MMVFW.MDL

mciGetPositionString

Syntax:	mciGetPositionString (<Multimedia component name>)
Description:	This function returns a string describing the current position of a multimedia component. The contents of this string depends on the currently selected multimedia device.
Parameter(s):	Name of a multimedia component.
Return Value(s):	A string describing the current position of the file in device specific format.
Example:	You may want to use this function to continuously display the position of a playing multimedia device by assigning the result of this function to a text component through a link on the multimedia event “Position Changed”.
See Also:	mciGetPosition
Category:	MMVFW.MDL

mciGetRepeat

Syntax:	mciGetRepeat (<Multimedia component name>)
Description:	This function determines if the given mulmultimedia component is in repeat mode. ReRepeat mode means that the multimedia comcomponent is played repeatedly. Repeat mode is set through

	a multimedia link com
	mand.mand.
Parameter(s):	Name of a multimedia component.
Return Value(s):	0 if not in repeat mode, 1 if in repeat mode.
Example:	`mciGetRepeat (mci1)` → `0` (not in repeat mode)
Category:	MMVFW.MDL

mciGetSpeed

Syntax:	mciGetSpeed (<Multimedia component name>)
Description:	This function returns the currently selected output speed of a multimedia component. The value is returned in percent of the normal output speed. Therefore 1000 means normal speed, 500 means half speed.
Parameter(s):	Name of a multimedia component.
Return Value(s):	Numerical value specifying the percentage of normal speed.
Example:	`mciGetSpeed (mci1)` → `1000`
Category:	MMVFW.MDL

mciGetStart

Syntax:	mciGetStart (<Multimedia component name>)
Description:	This function returns the start position of a multimedia component. The unit of this value depends on the currently selected multimedia device. (A

	Video for Windows file (*.AVI) usually returns zero).
Parameter(s):	Name of a multimedia component.
Return Value(s):	Numerical value specifying the start position.
Example:	`mciGetStart (mci1) → 0` ; where mci1 is an .AVI file.
See Also:	mciGetLength
Category:	MMVFW.MDL

mciGetVolume

Syntax:	mciGetVolume (<Multimedia component name>)
Description:	This function returns the sound volume of a multimedia component, if applicable. A value of 1000 specifies the normal output volume. Specify lower values to decrease the volume and higher values to increase the volume.
Parameter(s):	Name of a multimedia component.
Return Value(s):	Numerical value specifying the volume level.
Example:	`mciGetVolume(mci2) → 1000` ; where mci2 points to a .WAV file.
Category:	MMVFW.MDL

mciSendString

Syntax:	mciSendString (<text>,<text>,<Numerical

value>,<Numerical value>)

Description: This function is imported directly from the Windows Multimedia support DLL. It sends commands immediately to a multimedia device and returns the result string.
You may want to use this function to control media devices not supported by the multimedia component.

Parameter(s): The first string is the command to be sent to the multimedia device.
The second parameter should be the name of an input field which will receive the response to the command.
The third parameter actually specifies the maximum length of the response. It should be set to 255.
The fourth parameter must be 0.

Return Value(s): This function returns zero if it was successful. Otherwise, a device specific error code is returned. Please refer to the documentation of the multimedia device.

Example:
```
mciSendString ("play "
          +mciGetAlias(mci1),
          edt4, 255, 0)
```
will start to play the .AVI file referenced in mci1.

See Also: mciGetAlias

Category: MMPARSE.MNC

Registering External Functions

Following is an example of how mindmap can read and write strings to and from INI files.

Please verify that your MMPARSE.MNC file includes the following two lines and that the file MMPSTOOL.DLL is in your mindmap home directory, or at least in a path pointed to by the PATH environment variable.

```
declare WriteProfile lib 'mmpstool.dll' alias
'WriteProfile' (string, string, string, string)
as integer

declare ReadProfile lib 'mmpstool.dll' alias
'ReadProfile' (string, string, string, string) as
string

declare GetHomeDir lib 'mmpstool.dll' (string) as
string
```

The parameters are interpreted as follows:

ReadProfile

1st string	section name in the INI file.
2nd string	key name in the section.
3rd string	default return value if the section or the key are missing.
4th string	name of the INI file. If you omit the path in this string, the windows directory is assumed.

WriteProfile

1st string	section name in the INI file.
2nd string	key name in the section.
3rd string	value to write into the INI file.
4th string	name of the INI file. If you omit the path in this string, the windows directory is assumed.

GetHomeDir

1st string file name without path specification.

Now create a value assignment and select as the target, an input field or a text field to contain the name of the actual entry. Enter the following line into the value editor of the assignment message:

```
ReadProfile
('System','Test','C:\AUTOEXEC.BAT','DEMO.INI')

WriteProfile
('System','Test','C:\TEST.BAT','DEMO.INI')
```

If you want to place the INI file into the mindmap home directory, you may replace the specifier 'MINDMAP.INI' with GetHomeDir('MINDMAP.INI').

```
ReadProfile
('System','Test','C:\AUTOEXEC.BAT',GetHomeDir('DE
MO.INI'))

WriteProfile
('System','Test','C:\TEST.BAT',GetHomeDir('DEMO.I
NI'))
```

During load-time, mindmap reads the file MMPARSE.MNC which allows the registration of functions from user-supplied DLLs into the parser. Subsequently, such functions may be used in value assignment messages, conditions, and all other places where strings are parsed.

Please note that there are some restrictions related to the registration of external functions:

1. All external functions must follow the FAR PASCAL calling scheme. (This convention implies that parameters are pushed on the stack from left to right. This also means that variable length parameter lists are not available. The called function is responsible for cleaning up the stack. FAR implies that the function is called via a 32-bit address.)
2. Only four types of parameters are supported: ASCIIZ-String (string), 16-bit integer (integer) , 32-bit integer (long), and 8 byte floating point (double).
3. Integer, long and double are passed as 'call by value' if not otherwise declared. Numeric data types can be forced to follow the 'call by name' scheme by adding the byname modifier.

4. Strings are passed as 'call by name' as FAR. If a DLL function passes a string back to the caller, the buffer pointed to by the string is defined to hold al least4096 characters, including the terminating null. If the component (e.g. an input field) already contains a larger string this string is passed in its complete length. Remember that the called function cannot reallocate this pointer.

5. An external function may return either a 16-bit integer, a 32-bit integer or a FAR pointer to a string. In the latter case, the buffer pointed to by the return value must either exist in the default data segment of the called function or be allocated in local or global memory. It must not be allocated on the stack (automatic variable).

The following sample shows how to register the WinHelp function from the Windows Kernel:

```
declare WinHelp lib 'user' (integer, string, integer, long) as integer
```

The next example shows how a floating-point parameter can be passed back to mindmap:

```
declare SampFunc lib 'sample.dll' (integer, integer, double byname) as integer
```

The corresponding declaration in the C-source for sample.dll would then look like:

```
short __far __pascal __export SampFunc (short x, short y, double __far* pReturn);
```

Additionally, the system allows constants to be defined through the following syntax:

```
const SW_HIDE = 0
```

Please note that syntax errors in the MNC file are reported through entries in the MMERROR.LOG file, which may be examined by using the MMINFO utility in the Options/Preferences menu.

Also note that if mindmap detects an error in an MNC file the interpretation is abandoned.

Appendix

Glossary

3GL (third-generation language)	Traditional programming language like COBOL or FORTRAN.
4GL (fourth-generation language)	Successor to 3GLs like FORTRAN and COBOL. 4GLs are usually proprietary and require an average of 1/10th the code of 3GLs. 4GLs are becoming more object-based, but still remain procedural in nature and require serious dedication.
Access	Microsoft single-user desktop database product.
Alignment	The way text within a text object or input field is positioned: left (flush left, ragged right); right (flush right. Ragged left); centered (each line centered within the text object); or justified (flush right and flush left).
ANSI (American National Standards Institute)	The main standards body for U.S. computing.
API (application programming interface)	Specification or actual function library that provides a standard look to programs. There are lots of APIs, but only some are made public and become industry standards.
Arrange	A command on the Properties menu that aligns objects in precise relation to each other.

ASCII (American Standard Code for Information Interchange)	Seven-bit code for representing standard characters.
Attributes	Properties of a component that can be viewed and changed in the component's attribute toolbox, such as font, color, border, etc.
Background	(1) A design shared by other pages. Components on the background appear in the same position, style, and size on every page sharing that background. (2) The layer first created on a page.
BASIC (Beginner's All-purpose Symbolic Instruction Code)	3GL programming language that serves as the basis for Microsoft's Visual Basic and macro languages such as LotusScript.
Bitmap	A stored set of bits in computer memory that defines each pixel in an image.
Border	An attribute of some components.
Button	A component frequently used to enable user interaction.
C/ C++	C is a traditional procedural 3GL, but C++ is a very different object-oriented language that happens to have been built on C. Both are still the most widely used development languages in the PC and workstation market, although JAVA has made serious inroads during 1996.

Class	In object-oriented terminology , an abstract definition of a group of objects that have similar characteristics, including associated code and data structures. Individual instances of a class are objects. Classes are often organized in hierarchies based on inheritance. Class libraries typically contain several class hierarchies.
Cell	A single rectangle, the intersection of a row and a column in the worksheet, where one piece of data can be entered.
Class library	See Class.
Client	The PC or program that requests services from a server.
Client/ server	An architecture that divides work between the desktop and the back end server.
COBOL	COmmon Business-Oriented Language. The most common programming language on mainframes. Generally used for business applications.
Color palette	The set of colors mindmap has available. This is based on the settings in Windows and the installed hardware.
Column heading	A horizontal numbered button in the data table that identifies a column.
Compiler	A program which accepts source code (COBOL, C, FORTRAN, etc.) and translates it into machine language.

Component	A component is the fundamental building block in mindmap. It has (1) a general purpose, (2) a set of attributes, (3) events it reacts to, and (4) messages it can generate.
Copy	A command on the Edit menu that copies selected objects, data, or text and places the copy on the Clipboard.
Custom control	In Windows parlance, a custom-built control. VBXs are a popular category of custom controls designed to extend the Visual Basic environment.
Cut	A command on the Edit menu that removes selected components, data, or text from the current location and places the selection on the Windows Clipboard.
Data dictionary	Usually a table or group of tables that contain information about one or more particular databases. Data dictionaries are related to repositories and system catalogs and are said to contain metadata, that is, data about data.
Data type	Defines the kind of data associated with a given data field. Fields are small, meaningful chunks of data like last names, and data types describe the kind of data the fields contain, such as numeric, character, or date/time.
Database	Organized collection of facts, sometimes including documents and images.
DB2	IBM's popular relational DBMS for mainframes.

dBASE	Popular PC database program whose DBF file format has become an industry standard for sharing and transferring data. dBASE runs under DOS, Windows, Macintosh, UNIX, and VAX/VMS.
DBF	Published database file format originally developed for dBASE 1.1.
DBMS (database manageagement system)	Software that lets you create and maintain a database. Examples range from desktop DBMSs like Paradox and dBASE to enterprise DBMSs like Ora cle and SQL Server. ToOracle and SQL Server. Today's DBMSs are usually relational, meaning data is stored in spreadsheet-spreadsheet-like tables and related by common columns.
Debug	To find errors (bugs) in a program. Many application development products include debuggers for their own 4GL or script languages.
Default	A setting that mindmap automatically uses for commands and dialog boxes. General mindmap defaults can be changed using the File | Preferences command on the main menu. Component specific defaults are set by selecting any component and then setting the attribute via the main menu.

Delimiter	A single character, such as a tab, that separates each piece of data in a source file.
Dithering	A Windows process of blending colors to approximate a color. This process is required whenever the original bitmap has a palette larger than the one available on the system which is used to display it.
Drag&Drop	A mechanism by which a component offers its contents in various formats to other components, and vice versa a mechanism by which a component can receive data.
Duplicate	A command on the Edit menu that duplicates a selected object or slide and pastes the copy into the current presentation, bypassing the Windows Clipboard. mindmap also uses the **F2** function key for duplication.
EBCDIC (Extended Binary Coded Decimal interchange Code)	8-bit code used in IBM mainframes. Incompatible with the ASCII used in most PCs.
Edit mode	The mode in which applications are developed in mindmap.
Email (electronic mail)	The messages themselves or the software controlling their creation and routing.
Encapsulation	Characteristic of OOP whereby code and data are combined into a single entity known as an object that can be manipulated by a programmer without knowing the details of its implementation. mindmap encapsulates functionality by means of the encapsulation component.

Event	A trigger interpreted by receiving components
Event driven	A radical departure from traditional 4GL development environments, in which the approach is completely non-procedural.
Fill color	The color applied to the interior of an object or the part of an object that the assigned line color surrounds.
Font	The complete set of characters for one typeface, style, and size, such as Arial 10-point bold.
FORTRAN	3GL that's particularly effective for writing scientific and engineering programs. Stands for 'FORmula TRANslator.'
Front end	Sometimes used as a synonym of client to refer to the program running on a PC.
Full screen mode	The mode mindmap puts applications at run time into, in which standard screen elements such as the caption bar, system menus, etc. are not displayed. Most commonly used for multimedia applications.
GB	Gigabyte, or one billion bytes.
Gradient	The gradual blending of two or more colors. Gradients can be used to fill objects or enhance the background.
Grid	A set of dots used to precisely align objects on a slide. The grid does not show on the page itself.

Group	A command on the Properties menu that binds multiple components together so that they can be manipulated as one component.
GUI (graphical user interface)	Operating system programs, such as Microsoft Windows, X Windows MoMotif, or the Macintosh desktop, that use graphics and icons in addition to text. Compare to character-based terminals and operating systems such as MS-DOS.
Hz (Hertz)	Unit of frequency (per second) used to indicate computer processor speed, for example, 100 MHz.
Inheritance	Mechanism for organizing and maintaining a collection of classes that supports reusability by defining new classes (subclasses) in terms of existing ones. C++ and some GUI 4GL tools support multiple inheritance, but SmallTalk supports only single inheritance.
Insertion point	The I-beam that appears and indicates where typing can begin.
Instance	Same as object. Every object is an 'instance' of a particular class. Objects are 'declared' by instantiating them.

Integrity	A measure of a database's accuracy. There are several kinds of integrity, but in relational databases, one of the most important is referential integrity. Referential integrity means that related data that's split up into two or more tables will be kept in synch. For example, if you delete an order, referential integrity says you should delete all the order items also.
IP (Internet Protocol)	Standard internetwork routing protocol in the TCP/IP stack.
IPX (Internet Exchange Protocol)	Novell's packet assembly and routing protocol, corresponding more or less to the data link and network layers of the seven-layer OSI reference model.
ISDN (Integrated Services Digital Network)	An digital phone line that supports both data and voice simultaneously. An alternative to traditional analog phone lines.
join	A common process of combining data from one (self-join) or more relations in database tables, when performing a query or printing a report.
Label	Text that appears on a button or as a description of a component onscreen.
Layer	Stacking area of a page on which objects are created. If one component appears to be on top of another, the top component is on a layer closer to the front of the page.

Line color	The color applied to object borders, patterns, lines, text, and button labels.
Link	The pair of an incoming event and the outgoing message, as viewed from the perspective of a component.
Locking	An action initiated by a DBMS to ensure data integrity. If locking didn't occur, two users could be simultaneously updating the same record, unaware of the other's changes. Only the last person's changes would 'take.'
Mainframe	Large computer that generally costs several million dollars and shared by a large number of users simultaneously.
MAPI (Microsoft Mail API)	Part of Microsoft's Windows API particular to email.
Master	In relational databases, often used to describe master/detail relationships that reflect data stored in master and detail tables, for example, invoices and invoice detail tables.
Method	Function or procedure defined for a class. Most methods are associated with objects (defining what happens when a user clicks on push button, for example), but you also use methods for non visible objects, for example, that define error handling routines.
MB	Megabyte, millions of bytes. Unit of memory (RAM) or disk storage.
Notes (Lotus Notes)	Cross-platform groupware from Lotus Development Corporation.

Object	Basic building block of object oriented products, Objects are instances of (derived from) classes and often reflect a real-world object like employees. Objects, unlike simple variables in traditional programming , include both data and methods (methods are like functions or procedures). This bundling of data and methods is referred to as encapsulation. Anything on a page that can be selected, moved, or resized.
Object-oriented	Refers to programming languages. development tools, and methodologies based on objects, inheritance, polymorphism, and encapsulation.
OOP (object-oriented programming)	Considered the successor to 3GL procedural programming. C++ and SmallTalk are object oriented programming languages. Generally, a traditional 3GL programmer will need about six months to learn this new programming paradigm.
Palette	The set of all colors and custom colors from which a color can be assigned to a mindmap component.
Palette conflict	A conflict of colors that can occur when an animation or video's color palette contains enough colors that are different from MindMap's color palmindmap's color palette.

Parser	The formula interpreter built in to mindmap. It accepts statements, parses them, resolves references and proceeds to interpret them. It can be extended by mere declaration of the new functions in a text file. The parser has no influence on the actual flow of the application.
Paste	A command on the Edit menu that places the contents of the Windows Clipboard onto the current slide.
Pattern	A pixel-based fill pattern a developer can assign to some components.
Peer-to-peer	Said of LANs that don't require a central network server where the NOS is installed. In peer-to-peer LANs, all workstations have copies of the entire NOS.
Polymorphism	A characteristic of OOP that fosters reusability because it lets programmers use the same object with different methods.
Procedural programming	A style of programming equated with 3GLs and some 4GLs characterized by a sequential flow of instructions. Compare to event-based and object-oriented programming. mindmap is event-driven.

Protocol	Generic term that is often used as a synonym for the term standard. Network protocols, however, are the means for establishing communications between network devices. Leading network protocols include TCP/IP, DECnet, and IPX/SPX.
RAD (rapid application development)	Popular model for software development that focuses on prototyping ,d iterative development.
Referential integrity	Making sure that data in related tables is kept in synch.
Relational	Refers to the popular database model whereby data are stored in cables. SQL is the standard query language used to access data stored in relational databases like Oracle, Informix, and SQL Server.
Run mode	The mode in which mindmap executes applications.
SELECT	(1) The main command in SQL for retrieving data. For example, SELECT * FROM Vendors WHERE State = 'NY' would return complete record information (* is an abbreviation for ALL) from a user-defined database table called Vendors, but only records from vendors with the value NY in the state field. (2) The action of clicking an object with the selection arrow to display the object's selection handles. An object must be selected before it can be edited
Size	(1) The act of changing the shape or an object. (2) The point size assigned to text.

Socket	The specific address of a process running on a port (terminal or work station) connected to a TCP/IP LAN.
SQL (Structured Query Language)	A de facto standard query language for today's relational databases. Because SQL was designed to be a query, language, not a complete database language, most database vendors have added their own extensions,' which means doing distributed queries involving more than one DBMS call be particularly challenging.
Subassembly	A mindmap application (.MM file) which has exposed itself to other mindmap applications by means of the encapsulation component.
Table	Two-dimensional array for storing database records in relational databases. Common tables in business databases could be named Products, Vendors, Invoices, Invoice Detail, Suppliers, Purchase Orders, Purchase Order Details, Customers, and Employees. Each record should have a key field, such as customer number, in order to avoid ambiguity.
Transparent	An attribute assigned to a mindmap component that allows components on layers behind the transparent component to show through.
UNIX	Any of various related operating systems found primarily on mid-range systems and workstations,

VBA (Visual Basic for Applications)	A common scripting language supported by many Microsoft applications. Based on Visual Basic, but not exactly the same as it.
VBX	Special kind of Windows DLL associated with Visual Basic custom controls. Many commercial VBX add-ins are available and most work with Visual C++, too. mindmap supports VBX technology.
Visual Basic	A popular Windows based programming language from Microsoft.
Waterfall method	A five-step, sequential method for developing software that comprises these steps: fact gathering and application specification, analysis, design, coding, and testing.
Zoom tool	The magnifying glass or the tool palette that magnifies or reduces the view.

Entries in MINDMAP.INI

Section / Entry	Values	Default	Description
[System]			
Language			Name of a language DLL, such as MMENG.DLL. Since this DLL is assumed to be in the mindmap home directory, a path is not required.
HelpFile			Specifies the name of the default help file.
SampleFont	1 or 0	1	A value of 1 indicates that the sample font is to be displayed in the font selection dialog box.
Toolbox	x, y		Specifies the horizontal and vertical coordinate of the toolbox.
Messages	1 or 0	1	Enables the display of the status bar.
Bookmark	1 or 0	0	Enables the display of bookmarks in the status bar.
Grid	x, y	5, 5	Specifies the horizontal and vertical grid.
DuplicateOffset	x, y	5, 5	Defines the horizontal and vertical offset to which components are moved after they have been duplicated.
Marker	x	8	Defines the size of the 8 drag marks when a component has been selected.
Screen Mode	1 or 0	0	Reflects the setting for Disregard ...

Section / Entry	Values	Default	Description
Page Icons	1 or 0	1	Enables the page overview feature.
Page Icon Width	w	128	Defines the width of an iconized page.
Page Icon Height	h	96	Defines the height of an iconized page. Width and height are recommended to have a ratio of 4:3.
Import Directory			This entry specifies the default directory where graphic components are being loaded from in edit mode.
Import Format			The name of the least recently used graphic import format.
Working Directory			This entry specifies the directory which is used for locating mindmap applications, i.e. it is the default directory for the File | Open command.
Copy Links	0=No, 1=Yes, 2=Ask		Defines the operation mode for copying links when a component is being copied.
Edit Object Attributes	0=none, 1=LDBL, 2=RUP, 3=RDBL[1]	0	This entry specifies how to enter the component specific attributes dialog.
Default Attributes	0=none, 1=LDBL, 2=RUP, 3=RDBL	2	This entry specifies how to enter the attributes dialog.
Popup Menu	0=none, 1=LDBL, 2=RUP, 3=RDBL	130	This entry specifies how to popup the system menu.

[1] LDBL = Left mouse double clicked, RUP = Right mouse released, RDBL = Right mouse double clicked. For the Ctrl key to be used, add 128 to this value, for Shift add 64.

Section / Entry	Values	Default	Description
MinRectSize	x, y	5, 5	Specifies the minimum horizontal and vertical size of a component. Also, these values define how large a new component has to be drawn at least for being created.
Undo Buffer Size		40	Specifies how many actions can be undone.
Create Backup	n	3	Defines the number of backup copies of an application to be maintained.
Object Icons	1 or 0	1	Enables icons specific to each component class to be displayed in object lists.
ReadCache	1 or 0	1	Enables the read cache when loading mindmap applications.
Recover	1 or 0		Specifies if mindmap should attempt to recover damaged application files automatically.
PaintAsBitmap	1 or 0	0	This entry enables a mindmap page to be prepared in background before display. We recommend enabling this feature for Multimedia applications which require a smooth screen display.
Compression	1 or 0	0	Enables saving applications in compressed format.
LoggingLevel	0 to 5	4	Specifies the level of system log entries in the file MMERROR.LOG.
UsePalette	1 or 0		Enables the use of colors from a common uniform palette. Applicable only to 256 color mode.
ShowFocus	1 or 0		Enables dotted rectangles for button components that have the input focus.

Section / Entry	Values	Default	Description
MarkerColor	r,g,b	255,0,0	Defines the color of the hatched border when a component has been selected.
SysLogWindow	left, top, right, bottom		This entry defines the position of the System Log Window.
SysLogMode	0=hide, 5=show	0	Specifies the status of the System Log Window.
SysLog	1 or 0	0	Enables entries in MMERROR.LOG to be visible in the System Log Window.
FullScreenBackground	r,g,b	192,192, 192	This value specifies the color or the background which is not used of an application running in full screen mode. This is applicable for applications that execute on screen resolutions higher than they have been designed for.
CenterFullScreen	1 or 0		Specifies if a full screen application is to be centered on the screen if running on higher resolution displays.
CriticalError	1 or 0		Enables the interception of critical errors. mindmap will attempt to save the current application in the case of a GPF if this entry is enabled.
FilterCache		4	Specifies the number of images that mindmap will try to keep in its internal cache buffer. This value should be at least the number of images that an application will display concurrently on the same page.

Section / Entry	Values	Default	Description
AppRecalc	1 or 0		Enables the recalculation of all components in the application whenever a component changes its value. If disabled, only the components on the current page or its background page will be recalculated. For increased performance we strongly recommend to disable this feature.
LinkTool	left, top, right, bottom		This entry defines the size and position of the link toolbox (Icon below the pointer on the toolbox).
UpdateObjectList	1 or 0	1	Enables the object list to be refreshed on each change in the order of components on a page or when components are being created or removed.
[Extensions]			
Application			The extension which a mindmap application is associated with.
PrinterTemplate			The extension which a printer template is associated with.
[Files]			
File1..File99			Up to 99 file names which have been recently loaded in the development environment.
[Libraries]			This section controls how mindmap module files (*.MDL) are loaded.
Default			

Section / Entry	Values	Default	Description
Section			This entry defines the name of another user-defined section which is to be used instead of the Libraries section. This user supplied section has the same entries as Libraries except the Section entry. Use this feature for defining multiple sets of loading schemes for different types of applications.
Lib1..Libx			Multiple entries for module files to be loaded. Please note that the module MMBASE.MDL must always be loaded first, if required.
[Fonts]			
Default			Default font used for all components dealing with fonts, as well as for the status bar.
Formula			Font that is used in dialog input fields which are parsed at run time.
ObjectName			The font that is used for component lists.
[PaperType]			
<Values>	width, height		This section contains user-defined entries which specify the names and dimensions (in pixels) of pages sizes to be displayed in the page setup dialog for the output page component or the printer templates. Example: Standard Letter=2032,2794

Section / Entry	Values	Default	Description
[Colors]			
Color1..Color16			This section defines up to 16 colors that replace the two bottom lines of the color selection dialog and the line style dialog.
[Parser]			
Background	r,g,b	255,255, 255	The background color for all dialog input fields which are parsed at run time.
Errors	0=never, 1=always, 2=exit run mode	2	Determines if and when errors from evaluating parser statements are shown.
Window			Position of the parser error dialog.
Helper			Position and size of the parser function help window.

Table of Parser Functions

ANSI Character Set

Dec	Hex	Dec	Hex		Dec	Hex	
0	0	32	20		64	40	@
1	1	33	21	!	65	41	A
2	2	34	22	“	66	42	B
3	3	35	23	#	67	43	C
4	4	36	24	$	68	44	D
5	5	37	25	%	69	45	E
6	6	38	26	&	70	46	F
7	7	39	27	‘	71	47	G
8	8	40	28	(	72	48	H
9	9	41	29	)	73	49	I
10	A	42	2A	*	74	4A	J
11	B	43	2B	+	75	4B	K
12	C	44	2C	,	76	4C	L
13	D	45	2D	-	77	4D	M
14	E	46	2E	.	78	4E	N
15	F	47	2F	/	79	4F	O
16	10	48	30	0	80	50	P
17	11	49	31	1	81	51	Q
18	12	50	32	2	82	52	R
19	13	51	33	3	83	53	S
20	14	52	34	4	84	54	T
21	15	53	35	5	85	55	U
22	16	54	36	6	86	56	V
23	17	55	37	7	87	57	W
24	18	56	38	8	88	58	X
25	19	57	39	9	89	59	Y
26	1A	58	3A	:	90	5A	Z
27	1B	59	3B	;	91	5B	[
28	1C	60	3C	<	92	5C	\
29	1D	61	3D	=	93	5D	]
30	1E	62	3E	>	94	5E	^
31	1F	63	3F	?	95	5F	_

Dec	Hex		Dec	Hex		Dec	Hex	
96	60	`	128	80	□	160	A0	
97	61	a	129	81	•	161	A1	¡
98	62	b	130	82	,	162	A2	¢
99	63	c	131	83	*f*	163	A3	£
100	64	d	132	84	„	164	A4	¤
101	65	e	133	85	…	165	A5	¥
102	66	f	134	86	†	166	A6	¦
103	67	g	135	87	‡	167	A7	§
104	68	h	136	88	ˆ	168	A8	¨
105	69	I	137	89	‰	169	A9	©
106	6A	j	138	8A	Š	170	AA	ª
107	6B	k	139	8B	‹	171	AB	«
108	6C	l	140	8C	Œ	172	AC	¬
109	6D	m	141	8D	•	173	AD	-
110	6E	n	142	8E	□	174	AE	®
111	6F	o	143	8F	•	175	AF	¯
112	70	p	144	90	•	176	B0	°
113	71	q	145	91	‘	177	B1	±
114	72	r	146	92	’	178	B2	²
115	73	s	147	93	“	179	B3	³
116	74	t	148	94	”	180	B4	´
117	75	u	149	95	•	181	B5	µ
118	76	v	150	96	–	182	B6	¶
119	77	w	151	97	—	183	B7	·
120	78	x	152	98	˜	184	B8	¸
121	79	y	153	99	™	185	B9	¹
122	7A	z	154	9A	š	186	BA	º
123	7B	{	155	9B	›	187	BB	»
124	7C	|	156	9C	œ	188	BC	¼
125	7D	}	157	9D	•	189	BD	½
126	7E	~	158	9E	□	190	BE	¾
127	7F	•	159	9F	Ÿ	191	BF	¿

Dec	Hex		Dec	Hex	
192	C0	À	224	140	à
193	C1	Á	225	E1	á
194	C2	Â	226	E2	â
195	C3	Ã	227	E3	ã
196	C4	Ä	228	E4	ä
197	C5	Å	229	E5	å
198	C6	Æ	230	E6	æ
199	C7	Ç	231	E7	ç
200	C8	È	232	E8	è
201	C9	É	233	E9	é
202	CA	Ê	234	EA	ê
203	CB	Ë	235	EB	ë
204	CC	Ì	236	EC	ì
205	CD	Í	237	ED	í
206	CE	Î	238	EE	î
207	CF	Ï	239	EF	ï
208	D0	Đ	240	F0	ð
209	D1	Ñ	241	F1	ñ
210	D2	Ò	242	F2	ò
211	D3	Ó	243	F3	ó
212	D4	Ô	244	F4	ô
213	D5	Õ	245	F5	õ
214	D6	Ö	246	F6	ö
215	D7	×	247	F7	÷
216	D8	Ø	248	F8	ø
217	D9	Ù	249	F9	ù
218	DA	Ú	250	FA	ú
219	DB	Û	251	FB	û
220	DC	Ü	252	FC	ü
221	DD	Ý	253	FD	ý
222	DE	Þ	254	FE	þ
223	DF	ß	255	FF	ÿ

Index

A

B

C

D

E

F

G

H

I