Magic Nodes for MediaMonkey 3

Documentation by Rick Parsons, Bristol, England.

Manual edition: **1.7 - 21 Jun 2008**

The latest edition of this manual can be downloaded from

http://west-penwith.org.uk/misc/MagicNodes.pdf

Acknowledgements

... to Pablo Shmerkin, the original author of Magic Nodes and from whom most of the introduction chapter was derived.

... to Zvezdan Dimitrijevic, the author of the current "unofficial" version who converted it to MediaMonkey 3, fixed many bugs and added a lot of features. He also helped me to get this manual accurate.

... to the creators of MediaMonkey, without whom we would have no reason to exist.

... and finally to the many creators of Magic Nodes masks who have contributed to the discussion on the MediaMonkey forum.

The thread on the forum for discussion and development of this version of Magic Nodes can be found at

http://www.mediamonkey.com/forum/viewtopic.php?t=19168

Table of Contents

Installation

Download

... from http://solair.eunet.yu/~zvezdand/MagicNodes.htm

Although this version (1.7.6.2 at the time of writing) works with MediaMonkey 2.2+, that capability is not covered by this manual.

Instructions

- Make sure that you are logged in to an account that has write access to the **`scripts\auto`** folder within the **`MediaMonkey`** program folder (**`x:\Program Files\MediaMonkey`** in Windows XP where **`x:`** is your system disk). This will generally be an administrator account.
- These instructions have changed from version 1.7.6 onwards. If you have an earlier version of Magic Nodes installed then please first remove any files **`MagicNodes*.vbs`** file from the **`scripts\auto`** folder.
- Make sure that the downloaded file is named **`MagicNodes.mmip`**. If it has gained a **`.zip`** suffix then please rename it first.
- Double click the downloaded **`.mmip`** file. MediaMonkey will open (unless it is already open) and the script will install itself to the above mentioned folder. There is no need for a reboot or even a restart of MediaMonkey.
- A selection of sample demonstration masks are provided (copies are provided in an appendix to this manual). If there are existing masks that you have defined with a previous version then the installer will ask if you want to merge the sample masks into the list.
- If you have multiple accounts on the same computer using MediaMonkey then Magic Nodes will be available to all of them but each will have their own personal set of masks. Each account will be asked if they wish to merge the sample masks into their own library.

Important

If you have previously used older versions of Magic Nodes, particularly prior to version 1.6, then you should be aware that the syntax and function of many mask elements has changed. There is a detailed explanation at the start of the MediaMonkey forum thread mentioned in the acknowledgements above. All the latest information can be found there.

Uninstall

To uninstall the script, in MediaMonkey, go to **`Tools/Extensions`**, select **`Magic Nodes`** and click the **`uninstall`** button. Like installing, this must be done from a Windows Administrator account. The uninstall process will ask if you wish to also delete all the settings as well i.e. the defined masks. There is no need for a restart of MediaMonkey.

FAQ

These are copied directly from the forum mentioned above as they highlight problems that can be encountered. They mostly relate to advanced features so you may want to read them later.

Q. Is there a way that when starting MM3 the Magic Nodes tree is NOT expanded?

A. Install TweakMonkey (`http://www.mediamonkey.com/forum/viewtopic.php?t=8939`), Go to **`Tools/Options - TweakMonkey & Script Options/Tree Node Options`**, check the Node to be selected at startup, choose Library. Alternatively you could put all your Magic nodes into some new group instead of the Magic Nodes branch. You only need to put the **`Group`** pseudo-field in the front of each node and leave the Magic Nodes branch empty.

Q. If I ... (do something), the Magic Nodes nodes disappear. I can get it to reappear by creating a new Magic Node. Any ideas on what else to do to keep Magic Nodes visible?

A. You have most probably unchecked Magic Nodes in **`Tools/Options - Library/Filters and Views - None (Show all Tracks)`** then go to **`Configure/Default View Options`**. Alternatively there is a faster way to get the dialog box: right click on any node and select **`Options/Choose Tree nodes`**.

Q. I am getting the following errors when starting MM: Error #457 - This key is already associated with an element of this collection.

A. Most probably, your Custom X field has been renamed in such way to clash with existing field name. If this is a case, please try with some unique text value.

Q: How do I get the mask to skip unrated tracks and just make an average of rated ones? Right now it will only display averages for artists or albums which have all their tracks rated.

A: If there are **`sort by`** or **`statistic`** fields that are undefined, then the function is NOT applied. For example: **`<Album|Sort by:avg(Rating)>`** only displays the average rating for albums where ALL songs have been rated. For the others, just the album in alphabetical order is displayed. If you want to display these values even for those albums only have some tracks rated, currently the only solution is to manually hack the **`MagicNodes.vbs`** file. You should change the line that reads:

```
Const bSortStatOnlyIfPositive = True
```

to

```
Const bSortStatOnlyIfPositive = False.
```

Introduction

Nodes are the elements of the navigation tree structure of the MediaMonkey interface. The Magic Nodes script is a simple way of creating new ones to help you manage your music library. The Magic Nodes are defined using masks which are loosely based on the way MediaMonkey `auto-organize` works.

Getting Started

To create a node, use the `Edit => Create Magic Node` menu item. In order to create a node, you will need to specify a mask. Let us start with a simple example. Note: when practising, type the examples exactly as written, we will look at the detailed syntax rules later.

```
My First Magic Node\<Artist>\<Album>
```

This mask will create a node which is very similar to the Artist node that comes with MediaMonkey, so it's not very exciting yet (actually there are some differences, but we won't discuss them here). The node will be a child of the `Magic Nodes` root node, and its caption will be the first part of the mask (in this case, `My First Magic Node`).

Now, if you are like me you have too many artists so it would be convenient to organise them by their initial letter. This is not the way the standard Artist node works, but we can do a little magic with Magic Nodes. Let's create a new node with the following mask:

```
My Second Magic Node\<Artist|Trim:1>\<Artist>\<Album>
```

At this point, we will introduce some jargon. The stuff within the `< >` signs, representing some piece of information about your tracks, are called fields. So `Artist` and `Album` are two possible fields. In `<Artist|Trim:1>`, only `Artist` is the field; `Trim:1` is an instruction that configures the `Artist` field, and it is called a qualifier. Fields can be configured with any number of qualifiers, and qualifiers are separated by a `|` character.

The `Trim:` qualifier keeps only the specified number of characters from the field. So, `<Artist|Trim:1>` represents the first letter of the artist. Note that in the MediaMonkey `auto-organize` feature the same behaviour would be obtained with `<Artist:1>`. This is expanded in the first of the sample masks supplied in the Appendix.

Now, perhaps you want to organise by lyricist rather than artist. MediaMonkey provides no standard lyricist node. However, `lyricist` is one of many available fields in Magic Nodes, and you can use any of those fields to organise your tracks; a full list is included in the reference section later. The following mask will create a fancy lyricist node:

```
Lyricist|Child of:Artist\<Lyricist|Trim:1|Min tracks:10>\<Lyricist>\<Album>
```

We have introduced a number of new concepts here. Unlike the previous nodes, this one will show up right after the standard artist node in MediaMonkey. This is the result of including `Child of:Artist` in the mask. This is also a qualifier, but unlike `Trim:1` it affects the whole node, and not just one level. Such qualifiers are called global qualifiers, and are added after the node caption.

Notice the `Min tracks:10` qualifier. The effect is to ignore nodes with less than 10 tracks in your library. There's also a `Max tracks` qualifier available and you can use both.

At this point you may have many Magic Nodes, and you don't really want all of them. You can delete specific nodes by using the `Edit/Delete Magic Node` menu item; you'll have to lookup the mask in the sub-menu, or right click on the node itself where there is a `Delete Magic Node` option. Likewise, you can edit the mask of existing nodes by using the `Edit/Edit Magic Node` menu item or right click on the node (if you have used the `Child of:` qualifier you may need to restart MediaMonkey after deleting or editing; in that case, you will receive a warning message).

So far we have organised tracks according to the people involved in them. While there are still more related fields (like involved people), let us turn our attention to other kinds of fields.

You may have tracks in many different formats (mp3, wma, ogg, etc), encoded at different bitrates and sampling rates, some of them with VBR (Variable Bit Rate). There are many situations where being able to quickly browse your files by encoding may be useful. Also, notice that, although you can do an advanced search by bitrate, you can't search by either format or VBR. Here is how a typical encoding node would look (simplified from number 4 in the sample masks supplied):

```
Encoding|Shortcut:Ctrl+1\<Format>\<Bitrate>\<VBR>
```

Notice that we used a new global qualifier: `Shortcut:`. This is, hopefully, self-explanatory: after creating a node, pressing the keyboard combination `Ctrl+1` will expand the node and make it active (it is the same behaviour that `F8` has on the standard `Artist` node). You can specify any key combination which is not already used by Windows, MediaMonkey or other Magic Nodes.

Sorting

The built-in MediaMonkey nodes sort everything by alphabetical or numerical order. That is also the default behaviour for Magic Nodes. However, sometimes you may prefer to order, say, albums by year (and display the year at the same time), or by how popular they are with us (as measured either by the ratings or the number of times played). All this can be achieved with Magic Nodes; let us see how with another example:

```
Artist (by year)\<Artist>\<Album|Sort by:max(Year)|Sort order:desc>
```

This node is similar to our first Magic Node, except that it sorts the albums by year (showing the newest first); albums for which the year is unknown are shown at the end. In order to sort in some way other than the alphabetical album name, you use the `Sort by:` qualifier. Its use is a little bit involved though, because you also have to specify a function. To understand why this is the case, recall that the year can be assigned independently for every track, so it's possible that in one given album there are tracks with different years. So, a function must be specified to tell the script which year to use; in this case, `max()` was chosen (so, if there are tracks from different years, the most recent year is selected). If all the tracks in your albums have the same year information, then which function to use is largely irrelevant, but you still have to specify one. Other functions you can use include: `min()` and `avg()` (computes the average).

We have also used the `Sort order:` qualifier. There are two possible values for this qualifier: `asc` and `desc`, which, as you would expect, result in ordering the fields in ascending or descending order. By default, the order is ascending, so you need to include this qualifier if you want a descending order. You can use `sort order:` even if you are not using `sort by:`. For example, if you are organizing by `Bitrate` and you want to start with the highest bitrates, you can use the following mask:

```
Bitrate\<Bitrate|Sort order:desc>
```

Now suppose you want to see all of your albums ordered by popularity. Let's say, you want to sort them not alphabetically or by year, but according to how you rated them. In this case, it is sensible to use the `avg()` function (some albums have one or two highly rated tracks while the rest are bad; those will not appear at the top of your list). Here is how the mask may look (number 7 in the sample masks supplied):

```
Albums (by rating)\<Album|Sort by:avg(Rating)|Sort order:desc|Min tracks:5>
```

We have used the `Min tracks:` qualifier in a mask like this to see how complete albums performed. One feature of the `Sort by:` qualifier is that if the value (the rating, in our example) is not known for any track in the album, then the rating is not displayed (even if it's known for some tracks). Those albums are shown at the bottom of the list if the qualifier `Sort order:desc` was included, and at the top of the list otherwise.

Now assume you want to find out which albums you haven't listened to much in order to do some catching up. The field `played` stores the number of times a track has been played, and can be used within a `Sort by:` qualifier. Consider this example (number 8 in the sample masks supplied):

```
Scarcely played albums\<Album|Sort by:sum(Played)|Top:10|Min tracks:5>
```

This node will order your albums according to the number of times they have been played (which is the sum of the number of times each of their tracks have been played). Since no `Sort order:` qualifier has been specified, the default is used: ascending order. This is the desired behaviour because we want to find those albums which haven't been played much. There's also another qualifier we hadn't met before: `Top:`. This lets you show only a given number of values (albums, in this case).

Now you have got the general idea, we will look at the detailed specification of the script package which will show you all the possibilities available.

Reference

Mask Structure

`[Group specification\]*Caption[|Global qualifier]*\<Field[|Local qualifier]*>[\<Field[|Local qualifier]*>]*`

Where `[]` indicates optional and `*` indicates repeat 0-n times.

`Caption` is the caption you want displayed on the node. It can be a string containing any characters except `\` or `|`.

`Field` represent a hierarchical levels under the `Caption` and can be taken from a predefined list of tags and more complex combined fields listed below. At least one must be specified.

Each additional `\` indicates the start of a new hierarchical level in the tree.

The `Group specification` is an optional way to create a hierarchical structure of Magic Nodes in a similar way to the Media Monkey `Files to Edit` structure. It is important to note that this has no connection with the `Grouping` field name described below under Fields. The syntax is

Group specification ≡ <Group|Name:Caption[|Group qualifier]*>

`Group` is a pseudo Field, not derived from the library. See below for `Group qualifiers` and more details of this structure. As you can see the `Group specification` can be repeated to an arbitrary depth to create quite complex structures. Examples 28 to 50 in the samples supplied illustrate this in practice.

The case you use in the masks doesn't matter, All capitals or lower case or a mixture are accepted.

Spaces should not be included in mask definitions except ...

- `Caption`, `Field` names and `Qualifiers` which contain spaces; and then there should be just one space where required.
- In filter expressions where they are encouraged to aid readability.
- After the `:` of a `Qualifier` except in certain cases (as noted) where they are significant.
- Between functions (after the comma) in `Sort by:` and `Statistic:` qualifiers, and around the sort order in the `Sort by:` qualifier (if used).

Single spaces only should be used, not tabs or other white space. In particular, spaces are not required or allowed before the `:` of a `Qualifier`, around the `\` of a hierarchy level, the `|` separating the qualifiers, or the `<` and `>` of a `Field name`. See the detailed descriptions below for what these particulars mean.

Fields

Field	From		Description
`Album`	Songs.Album		Text. The title of the album.
`Album Artist`	#		Text. When ordering nodes, any leading small word prefix such as `The`, `A`, `Le` etc. is ignored. The list of prefxes can be found and altered at menu `Tools/Options – Library – Appearance` as a comma separated list.
`Album Volume`	Songs.NormalizeAlbum	%	A number in dB.
`Album Artist with Album`	`<Album Artist>` & `<Album>`		`<Album Artist> - <Album>`. Similar to `Album and Year` in an earlier version of Magic Nodes.
`Album with Album Artist`	`<album>` & `<Album Artist>`		`<Album> (<Album Artist>)`. Used to be `Album and Artist` in an earlier version of

			Magic Nodes.
Artist	#		The track artist. When ordering nodes, any leading prefix is ignored as for **Album Artist**.
Bitrate	Songs.Bitrate	%	Rounded to multiples of 16kbps when displayed as a node. As a **Statistic:** it is rounded to 1kbps but in a **Filter:** expression it is exact bits per second.
BPM	Songs.BPM	%	Beats Per Minute. It has the value -1 when unknown.
Channels	Songs.Stereo		The text **Mono**, **Stereo** or **Surround** both as a node and in an expression.
Comment	Songs.Comment		(not **Comments**! As some earlier documentation stated) The first line only is returned.
Composer	#		Text.
Conductor	#		Text.
Copyright	Songs.Copyright		Text.
Cover Storage	Covers.CoverStorage		The text **Tag** or **Image file** both as a node and in an expression. It does not distinguish between own or shared folders
Cover Type	#		Various wordy values are displayed and it can have multiple values for each track. In an expression each takes one of 21 different numeric values for example, 3 = Front cover, 4 = Back cover etc. A full list can be found at http://www.mediamonkey.com/wiki/index.php/Scripting_Resources#Cover_Types
Custom 1	Songs.Custom1	%	If your custom fields have been renamed in MediaMonkey then you can use the new user defined names but beware of unexpected results or errors if your name clashes with an existing field name in these tables. It is also recommended that you avoid names containing SQL reserved words such as AND which could make expressions ambiguous. They are returned as text in all contexts.
Custom 2	Songs.Custom2	%	
Custom 3	Songs.Custom3	%	
Custom 4	Songs.Custom4	%	
Custom 5	Songs.Custom5	%	
Date Added	Songs.DateAdded		As the text **yyyy-mm-dd**
Date Last Played	Songs.LastTimePlayed		As the text **yyyy-mm-dd**
Date Modified	Songs.FileModified		As the text **yyyy-mm-dd**
Days since added	Songs.DateAdded	%	Number.
Days since last played	Songs.LastTimePlayed	%	Number.
Days since modified	Songs.FileModified	%	Number.
Disc Number	Songs.DiscNumber	%	Returned as text but it works best if it is all numeric, alpha values could disrupt functions.
Drive Type	Songs.CacheStatus, Medias.DriveType and Medias.isAudioCD		Possible text values are **HD**, **Audio CD**, **CD/DVD**, **Virtual CD**, **Other removable media**, **RAM disk**, **Network** and **Other**.
Encoder	Songs.Encoder		The software used to encode the track.
File name	Songs.SongPath		The file name on the disk excluding the folder path and extension.
File size	Songs.FileLength	%	This is a number displayed as **nn to nn+1 MB** when a node, as a number of GB or MB when a **Statistic:** but in bytes when used in an expression.
Folder	Songs.SongPath		The folder path on the disk excluding any drive letter and file name. It includes a

			trailing \.
Format	Songs.SongPath		Text derived from the file extension.
Genre	#		Text.
Grouping	Songs.GroupDesc		This has no connection with the **Group** pseudo filed described above but is the MediaMonkey track property of that name.
Involved people	Songs.InvolvedPeople		Text. Not a multiple item field.
ISRC	Songs.ISRC		International Standard Recording Code.
Length	Songs.SongLength	%	This is displayed as **nn to nn+1 minutes** as a node but as [[hours:]minutes:]seconds as a **Statistic:**. When used in an expression it is a numeric value in milliseconds.
Leveling	Songs.NormalizeTrack	%	A number in dB.
Lyricist	#		Text.
Lyrics	Songs.Lyrics		Can only be used as a **Filter:** variable, not as a node.
Months since added	Songs.DateAdded	%	Number.
Months since last played	Songs.LastTimePlayed	%	Number.
Months since modified	Songs.FileModified	%	Number.
Mood	#		Text.
Occasion	#		Text.
Original Artist	Songs.OrigArtist		Text. When ordering nodes, any leading prefix is ignored as for **Album Artist**.
Original Lyricist	Songs.OrigLyricist		Text.
Original Title	Songs.OrigTitle		Text.
Original Year	Songs.OrigYear	%	Number.
Path	Songs.SongPath		The full path of the file containing the track which includes the folders, file name and extension, but not the drive letter. Be careful using this one as a node without modification as it will create a record for every track on the system which will be very slow.
Played	Songs.PlayCounter	%	Number as **nn times** or **not played** when used as a node but a plain number when used in a statistic or filter.
Playlist	#		Non-auto playlists only.
Play Rate	Songs.LastTimePlayed	%	Average number of days between plays. In a node this is displayed as **nn to nn+1 days**.
Publisher	Songs.Publisher		Text.
Quality	#		Text.
Rating	Songs.Rating	%	This is a number displayed in "Stars" from 0 to 5 in half units when used as a node or as a **Statistic:** but a value between 0 and 100 when used in an expression (-1 when unknown). The relationship is:-

Songs.Rating	Stars
-1	Unknown
0-5	0 (Bomb)
6-15	½
16-25	1
26-35	1½
36-45	2

46-55	2½
56-65	3
66-75	3½
76-85	4
86-95	4½
96-100	5

Sample Rate	Songs.SamplingFrequency	%	A number in kHz when displayed as a node or Statistic: but in Hz when used in an expression.
Tempo	#		Text.
Time since last played	Songs.LastTimePlayed		A wordy description in a node but when used in a filter it is the time in seconds. When used as a Statistic: they are grouped with values 0-60 in seconds, 61-3600 in minutes, 3601-86400 in hours and higher values in days.
Title	Songs.SongTitle		Text.
Track Number	Songs.TrackNumber	%	Text but it works best if it is all numeric, alpha values could disrupt functions. It can only be used as a filter, a Sort by: or Statistic: variable, not as a node.
VBR	Songs.VBR		Variable Bit rate displayed as CBR, VBR or N/A when used in a node but with a value of 0, 1 or -1 respectively when used in an expression.
Weeks Since Added	Songs.DateAdded	%	Number.
Weeks Since Last Played	Songs.LastTimePlayed	%	Number.
Weeks Since Modified	Songs.FileModified	%	Number.
Year	Songs.Year	%	Number.
Years Since Added	Songs.DateAdded	%	Number.
Years Since Last Played	Songs.LastTimePlayed	%	Number.
Years Since Modified	Songs.FileModified	%	Number.

Those marked % can be used with Sort/Statistic functions and/or numeric filters. Those marked # are multi-item fields.

Global Qualifiers

Child of:**Magic Nodes**|Album|Artist|Classification|Files to Edit|Genre|Library|Location|My Computer|Now Playing|Playlists|Previews|Net Radio|Rating|Virtual CD|Web|Year	Where to display the new node. Magic Nodes indicates the dedicated top level hierarchy, the others are in the standard MediaMonkey structure and Child of: means "after". Note that MediaMonkey 3 allows you to move nodes up and down using the Choose Tree nodes menu which can partially override this setting.
Icon:**Standard**|Top level|Bottom level|nn	Defines which icon from the field hierarchy to use for the caption node. Standard uses a folder icon. Alternatively a value of 0 to 57 representing icons in the list at http://www.mediamonkey.com/wiki/index.php/Scripting_Resources#Icons
Filter:Expression	See below for details of expressions.
Position:Before|After|First child|Last child	Where the node is placed in relation to other children. When Child of:Magic Nodes is used then Last child is the default and First child is the only other option available. When used with the Child of: options Location, Artist, Album, Genre, Year,

	`Rating`, `Virtual CD`, `Previews`, `Playlists` and `My Computer` then `After` is the default and `Before` is the only other option available. When used with `Child of:` options `Library`, `Classification`, `Files to Edit`, `Now Playing`, `Net Radio` and `Web` then `After` is the default and all other values can be used.
`Shortcut:Key combination`	Example: `Shortcut:Ctrl+F4`.
`Show tracks:`**`yes`**`|no`	Controls whether clicking on a field shows all the tracks at the this hierarchy level.
`SQL Filter:SQL expression`	See some examples below.
`Statistic` or `Statistics:Function [, Function]*`	The functions are listed below. The statistic is displayed after the field it qualifies in parentheses. Optionally multiple functions can be used; all are displayed. This is the same as the local qualifier mentioned below but beware, it can have significant performance issues if used as a Global qualifier.

Where **`bold`** indicates the default setting and `|` indicates possible alternate values.

Filter expressions

Spacing is permissible in the expression for readability. Text values should be enclosed in ‘single quotes’, field names should be enclosed in <angle brackets> (Note: this is different to earlier versions of Magic Nodes). They cannot have qualifiers applied to them.

The expression can consist of

conditionals: `=`, `<>`, `<`, `>`, `<=`, `>=` e.g. `<Field> > Value`

`<Field> between Value and Value`

`<Field> in (List of values)`

Expressions can be separated by booleans (`and, or`)

In fact, the syntax is much richer than this and can encompass all of the SQL Filter expressions described below, increasing their power considerably at the expense of complexity and possible portability problems if database tables are fields are referenced as the MediaMonkey database evolves.

Examples

```
Favourite tracks|Filter:<Rating> >= 85\<Title>

Recent albums|Filter:<Days since added> < 14\<Album>

Classical by Composer|Filter:Genre in ('Classical', 'Opera',
'Operetta')\<Composer>\<Album with Album Artist>

Best of the '70s|Filter:<Rating> >= 65 and <Year> between 1970 and 1979\<Album>

Modified today|Filter:<Days Since Modified> = 0\<Title>

Modified last week|Filter:<Days Since Modified> Between 0 And 6\<Title>

Modified on 2007-12-21|Filter:<Date Modified> = '2007-12-21'\<Title>
```

SQL expressions

MediaMonkey 3 uses SQLite so any SQL expression valid in that, and using the existing database fields, is permissible. Note that MediaMonkey 2 used a different engine and some different field names so these earlier expressions may no longer work.

The expressions should be exactly same as in the `WHERE` part of SQLite `SELECT` queries. They have a similar syntax to Filter expressions but need `table.field` names rather than Magic Nodes names for MediaMonkey fields. For example, if you want to use the `SQL Filter:` you could write

```
Songs.Artist = 'Pink Floyd'
```

but if you want to use the `Filter:` qualifier you could write

```
<Artist> = 'Pink Floyd'
```

Filter expressions are, in many cases, shorter than SQL expressions and are more likely to be independent of MediaMonkey revisions.

To discover the full power of SQL Filter expressions see an SQLite reference manual such as http://www.sqlite.org/lang.html

Examples

If you always download Internet tracks into a folders called "download" then this node could be useful

```
Downloaded tracks|SQL Filter:Songs.SongPath LIKE '%download%'\<Title>
```

```
Artists with more then one album|SQL filter:Songs.Artist IN (SELECT Songs.Artist
FROM Songs, ArtistsSongs, Artists WHERE Songs.ID = ArtistsSongs.IDSong AND
ArtistsSongs.IDArtist = Artists.ID AND Songs.IDAlbum > 0 AND
ArtistsSongs.PersonType = 1 GROUP BY Artists.ID HAVING Count(DISTINCT
Songs.IDAlbum) > 1)\<Artist>\<Album>
```

Note that the `SQL Filter:` qualifier can only take SQL expressions whereas the `Filter:` qualifier can take either form or a mix of the two. SQL filters as a separate qualifier may be deprecated in future releases.

Local Qualifiers

`All:`**yes**`|no`	Controls whether a node with all possible tracks is displayed at a this hierarchy level.
`Exclusive left of` or `Ex left of:Chars`	Display all characters to the left of the specified characters. If the characters do not appear in the value at all then a null (blank) value is returned.
`Exclusive right of` or `Ex right of:chars`	Display all characters to the right of the specified characters If the characters do not appear in the value at all then a null (blank) value is returned.
`Exclusive right until:chars`	Use with `Right of:` or `Exclusive right of:` to delimit the right hand end of the result.
`Left of:Chars`	Display all characters to the left of the specified characters. If the characters do not appear in the value at all then the whole value is returned.
`Max tracks:n`	Only display if there are n or less tracks below this level.
`Min tracks:n`	Only display if there are n or more tracks below this level.
`Right of:Chars`	Display all characters to the right of the specified characters. If the characters do not appear in the value at all then the whole value is returned.
`Right until:Chars`	Use with `Right of:` or `Exclusive right of:` to delimit the right hand end of the result. The examples below should make these qualifiers clear.
`Show sort key` or `Sort key:n`	Specify which sort by field is displayed. Default: 1. Value 0 indicates not to show any key (see examples below).
`Show tracks:`**yes**`|no`	The same function as the global qualifier above but at lower hierarchy levels.
`Sort by:Function [Order] [, Function [Order]]*`	The functions are listed below. The value to be sorted by is displayed before the field it qualifies, separated by a hyphen. Optionally the sort `Order` can be put after the function definition. Optionally multiple functions can be used (see examples below).
`Sort order:`**asc**`|desc`	Ascending or descending.
`Statistic` or `Statistics:Function [, Function]*`	The functions are listed below. The statistic is displayed after the field it qualifies in parentheses. Optionally multiple functions can be used; all are displayed.
`Top:n`	Only show the first of eligible items are listed. The `n percent` value is not supported in MediaMonkey 3.
`Trim:n`	Maximum number of characters to display from the left of the value. If `n` is negative then from the right.
`Unknown:`**yes**`|no`	Controls whether a node with unknown values is displayed at this hierarchy level.

Where **bold** indicates the default setting, `|` indicates possible alternate values, `[]` indicates optional and `*` indicates repeat 0-n times. Note that `Chars` in the `Left/Right` qualifiers should not have a leading space unless that is required to be part of the delimiting string and also should not contain any of the characters `\`, `|`, `<` or `>`.

Examples

If we consider a library containing album titles as in the first column below, the effect of various qualifiers is shown in subsequent columns. You can see that column three usefully extracts the composer, column five the Orchestra and column six the work. Note that I have used _ here to mark a significant space character.

`<Album>`	`Left of::`	`Exclusive left of::`	`Right of:-_`	`Exclusive right of:-_`	`Right of::_|Right until:_-`
Elgar: Enigma Variations - BBC Philharmonic	Elgar	Elgar	BBC Philharmonic	BBC Philharmonic	Enigma Variations
Gourmet Baroque - Academy of St. Martin in the Fields	Gourmet Baroque - Academy of St. Martin in the Fields	<Unknown>	Academy of St. Martin in the Fields	Academy of St. Martin in the Fields	Gourmet Baroque
Rossini: Stabat Mater	Rosini	Rossini	Rossini: Stabat Mater	<Unknown>	Stabat Mater

So using this, we can break out the titles in a classical music collection where they are in the form `[Composer]:_[Work]_-_[Orchestra]` using the mask

```
Classical Works\<album|Exclusive left of::>\<album|Right of::_|Right until:_-
>\<album|Exclusive right of:-_>
```

noting that the _ should be replaced by spaces. See samples 14 and 15 for another example of the use of these qualifiers.

Functions

`avg(Field)`	Valid fields are as listed above marked with %.
`count(All|Items|Tracks)`	`Items` is the number of node sub items below this level, `Tracks` is self explanatory. `All` is the same as `Tracks` but displays "`files`" in the caption. `count(Items)` can only be used for the `Statistic:` qualifier.
`max(Field)`	Valid fields are as listed above marked with %.
`min(Field)`	Valid fields are as listed above marked with %.
`sum(Field)`	Valid fields are as listed above marked with %.

The `First` and `Last` functions are currently not working in MediaMonkey 3. Note that, unlike for Filters, the field names are not in angle brackets.

Examples

```
Albums by average song rating\<Album|Sort by:avg(Rating)>
```

```
Albums by earliest song\<Album|Sort by:min(Year)|Show sort key:0>
```

```
Least played albums first\<Album|Sort by:sum(Played)>
```

Example using multiple sort functions

```
... Sort by:count(All) asc, max(Length) desc
```

`... Sort by:max(Custom 1), max(Year)|Show sort key:2`

An example of the use of `Statistic`:

```
Rating Genre\<rating|sort order:desc|statistic:count(all)>\<genre|sort
by:count(all)|sort order:desc>\<album artist|statistic:count(all)>\<Album>
```

Example using multiple statistic functions

`... Statistic:count(All), sum(File size), sum(Length)`

To mimic the old Album and Year field, try

`Album and Year\<Album Artist with Album|Statistic:max(Year)>`

Note: If there are sort fields that are undefined, then the function is NOT applied. (Example: `<Album|Sort by:avg(Rating)>` only displays the average rating for albums where ALL songs have been rated. For the others, just the album in alphabetical order is displayed.)

Group Qualifiers

`Name:Caption`	A **required** qualifier for `Group` which labels the node.
`Child of:`**`Magic Nodes`**`|Album|Artist|Classification|Files to Edit|Genre|Library|Location|My Computer|Now Playing|Playlists|Previews|Net Radio|Rating|Virtual CD|Web|Year`	As for the Global qualifier. Only the first specified `Child of:` qualifier in any one group takes effect.
`Icon:`**`Standard`**`|Top level|Bottom level|nn`	As for the Global qualifier.
`Position:Before|After|First child|Last child`	As for the Global qualifier. Only the first specified `Position:` qualifier in any one group takes effect.
`Show tracks:`**`yes`**`|no`	As for Global qualifier, controls whether clicking on a field shows all the tracks at the this hierarchy level.

Where **`bold`** indicates the default setting and `|` indicates possible alternate values.

Examples

```
<Group|Name:Problematic tracks|Show tracks:No|Child of:Files to Edit|
Position:Last child>\Tracks without lyrics|Filter:<Lyrics> = ''\<Title>
```

The group caption `Problematic tracks` represents first level of hierarchy inside of the `Files To Edit` branch and the caption `Tracks without lyrics` represents the second level. Note that the caption for THIS node (`Tracks without lyrics`) does not require angle brackets.

```
<Group|Name:Problematic tracks|Show tracks:No|Child of:Files to Edit|
Position:Last child>\Tracks without comment|Filter:<Comment> = ''\<Comment>
```

Because the `Group Name:` caption is the same, this node is in the same group as the one above but also note that the `Child of:` and `Position:` qualifiers also must match else it would be ambiguous. If `Show tracks:No` was not specified in these then the node at the top level would show all tracks that satisfied either filter criteria from the level below. Be warned, this could be very slow.

The Sample Masks

A copy of these masks can be found in \Program Files\MediaMonkey\Scripts\Auto\MagicNodes.ini

Note that many don't just work automatically but require you to tag your library in particular ways to provide the required information.

1. `Album Artist [A-Z], Album (by Year)|Icon:Top level\<Album Artist|Trim:1|Statistic:Count(All)>\<Album Artist|Statistic:Count(All)>\<Album|Sort by:Max(Year)|Statistic:Count(All)>`
2. `Genre, Artist, Album (with number of items)|Icon:Top level\<Genre|Statistic:Count(Items)>\<Artist|Statistic:Count(Items)>\<Album|`

`Statistic:Count(All)>`

3. `Rating, Genre, Album Artist with Album|Icon:Top level\<Rating|Sort order:Desc|Statistic:Count(All)>\<Genre|Sort by:Count(All)|Sort order:Desc>\<Album Artist with Album|Statistic:Count(All)>`
4. `Format, VBR, Bitrate|Icon:Top level\<Format|Statistic:Count(All)>\<VBR|Statistic:Count(All)>\<Bitrate|Statistic:Count(All)>`
5. `Cover storage|Icon:Top level|Show tracks:No\<Cover storage|Unknown:No|Statistic:Count(All)>\<Album>`
6. `Album statistics [A-Z]|Icon:Top level\<Album|Trim:1|Statistic:Count(Items)>\<Album|Statistic:Count(Tracks), Sum(File size), Sum(Length)>`
7. `Albums (by Rating)|Icon:Top level\<Album|Sort by:Avg(Rating)|Sort order:Desc|Min tracks:5>`
8. `Scarcely played albums|Icon:Top level\<Album|Sort by:Sum(Played)|Top:10|Min tracks:5>`
9. `Recently added Albums|Icon:Top level|Filter:<Weeks since added> < 10\<Album|Statistic:Avg(Days since added)>`
10. `Best of the '70s|Icon:Top level|Filter:<Rating> >= 65 and <Year> Between 1970 And 1979\<Album with Album Artist>`
11. `Top 10 rated Artists|Icon:Top level\<Artist|Sort by:Avg(Rating)|Sort order:Desc|Top:10>`
12. `One-hit wonders|Icon:Top level\<Artist|Max tracks:5>`
13. `High PlayCount or Rating Few Tracks|Icon:Top level|SQL Filter:(Songs.PlayCounter > 20 OR Songs.Rating > 60) AND Songs.ID IN (SELECT Songs.ID FROM Songs, ArtistsSongs, Artists WHERE Songs.ID = ArtistsSongs.IDSong AND ArtistsSongs.IDArtist = Artists.ID AND ArtistsSongs.PersonType = 1 GROUP BY Artists.ID HAVING Count(*) < 10)\<Artist>`
14. `Featuring artist|Icon:Top level\<Artist|Unknown:No|Statistic:Count(All)|Ex Right of: feat. |featuring |vs. |pres. |presents >`
15. `Main artist (only with collaborations)|Icon:Top level\<Artist|Unknown:No|Statistic:Count(All)|Ex Left of: feat. |featuring |vs. |pres. |presents >`
16. `Artist does not equal Original Artist|Icon:Top level|Filter:UpperW(Songs.Artist) <> UpperW(<Original Artist>)\<Artist>`
17. `Artists with more than 5 words in the name|Icon:Top level|SQL Filter:Songs.ID IN (SELECT Songs.ID FROM Songs, ArtistsSongs, Artists WHERE Songs.ID = ArtistsSongs.IDSong AND ArtistsSongs.IDArtist = Artists.ID AND ArtistsSongs.PersonType = 1 AND Length(Artists.Artist) - Length(Replace(Artists.Artist, ' ','')) > 4)\<Artist>`
18. `Artists with more then one album|Icon:Top level|SQL filter:Songs.Artist IN (SELECT Songs.Artist FROM Songs, ArtistsSongs, Artists WHERE Songs.ID = ArtistsSongs.IDSong AND ArtistsSongs.IDArtist = Artists.ID AND Songs.IDAlbum > 0 AND (ArtistsSongs.PersonType = 1 OR ArtistsSongs.PersonType IS NULL) GROUP BY Artists.ID HAVING Count(DISTINCT Songs.IDAlbum)> 1)\<Artist|Statistic:Count(Items)>\<Album>`
19. `Artists with different Genres|Icon:Top level|SQL Filter:Songs.Artist IN (SELECT Songs.Artist FROM Songs, ArtistsSongs, Artists, GenresSongs, Genres WHERE Songs.ID = ArtistsSongs.IDSong AND ArtistsSongs.IDArtist = Artists.ID AND Songs.IDAlbum > 0 AND (ArtistsSongs.PersonType = 1 OR ArtistsSongs.PersonType IS NULL) AND Songs.ID = GenresSongs.IDSong AND GenresSongs.IDGenre = Genres.IDGenre GROUP BY Artists.ID HAVING`

Count(DISTINCT Genres.IDGenre) > 1)\<Artist>\<Genre>

20. Tracks with track leveling, but no album leveling|Icon:Top level|Filter:Songs.NormalizeTrack > -999999.0 AND Songs.NormalizeAlbum = -999999.0\<Album>
21. Tracks with similar Comment|Icon:Top level|SQL filter:substr(Songs.Comment, 1, 4) IN (SELECT substr(Comment, 1, 4) FROM Songs GROUP BY substr(Comment, 1, 4) HAVING Count(*) > 1)\<Comment|Trim:4>
22. Tracks with same Name from same Artists|Icon:Top level|SQL Filter: Songs.SongTitle || '@#$' || Songs.Artist IN (SELECT SongTitle || '@#$' || Artist FROM Songs WHERE Length(SongTitle) > 0 GROUP BY SongTitle, Artist HAVING Count(*) > 1 AND Count(DISTINCT Artist) = 1)\<Title|Trim:1>\<Title>\<Artist>
23. Tracks with same Name from different Artists|Icon:Top level|SQL Filter: Songs.SongTitle IN (SELECT SongTitle FROM Songs WHERE Length(SongTitle) > 0 GROUP BY SongTitle HAVING Count(DISTINCT Artist) > 1)\<Title|Trim:1>\<Title>\<Artist>
24. Albums with Same Name|Icon:Top level|SQL Filter: Songs.IDAlbum IN (SELECT DISTINCT Albums.ID FROM Albums INNER JOIN Albums As Inline ON Albums.Album = Inline.Album AND Albums.ID <> Inline.ID)\<Album with Album Artist>
25. Complete Albums|Icon:Top level|SQL filter: Songs.IDAlbum IN (SELECT IDAlbum FROM Songs GROUP BY IDAlbum HAVING Count(TrackNumber) = Max(Cast(TrackNumber As integer)) AND Count(TrackNumber) > 3)\<Album|Statistic:Count(All)>
26. Incomplete Albums|Icon:Top level|SQL filter: Songs.IDAlbum IN (SELECT IDAlbum FROM Songs GROUP BY IDAlbum HAVING Count(TrackNumber) <> Max(CAST(TrackNumber AS integer)) AND Count(TrackNumber) > 1)\<Album|Statistic:Count(Tracks), Max(Track Number)>
27. Compilations|Icon:Top level|SQL Filter: Songs.IDAlbum IN (SELECT IDAlbum FROM Songs WHERE IDAlbum > 0 GROUP BY IDAlbum HAVING Count(DISTINCT Artist) > 1)\<Album|Statistic:Max(Year)>
28. <Group|Name:Albums with multiple...|Show tracks:No>\Bitrates|Icon:Bottom level|SQL Filter:Songs.IDAlbum IN (SELECT IDAlbum FROM Songs WHERE IDAlbum > 0 AND Songs.VBR = 0 GROUP BY IDAlbum HAVING Count(DISTINCT Round(Bitrate / 1000)) > 1)\<Album>\<Bitrate|Statistic:Count(All)>
29. <Group|Name:Albums with multiple...|Show tracks:No>\Years|Icon:Bottom level|SQL Filter:Songs.IDAlbum IN (SELECT IDAlbum FROM Songs WHERE IDAlbum > 0 AND CAST(substr(Songs.Year, 1, 4) AS integer) > 1900 GROUP BY IDAlbum HAVING Count(DISTINCT CAST(substr(Songs.Year, 1, 4) AS integer)) > 1)\<Album>\<Year|Statistic:Count(All)>
30. <Group|Name:Albums with multiple...|Show tracks:No>\different Genres|Icon:Bottom level|SQL Filter:Songs.IDAlbum IN (SELECT IDAlbum FROM Songs, GenresSongs, Genres WHERE IDAlbum > 0 AND Songs.ID = GenresSongs.IDSong AND GenresSongs.IDGenre = Genres.IDGenre GROUP BY IDAlbum HAVING Count(DISTINCT Genres.IDGenre) > 1)\<Album>\<Genre|Statistic:Count(All)>
31. <Group|Name:Albums with multiple...|Show tracks:No>\multi-item Genres|Icon:Bottom level|SQL Filter:Songs.IDAlbum IN (SELECT IDAlbum FROM Songs WHERE IDAlbum > 0 GROUP BY IDAlbum HAVING Count(DISTINCT Genre) > 1)\<Album>\<Genre|Statistic:Count(All)>
32. <Group|Name:Albums with multiple...|Show tracks:No>\FileTypes|Icon:Bottom level|SQL Filter:Songs.IDAlbum IN (SELECT IDAlbum FROM Songs WHERE IDAlbum > 0 GROUP BY IDAlbum HAVING Count(DISTINCT Upper(CASE WHEN substr(Songpath, -3, 1) = '.' THEN substr(Songpath, -2, 2) WHEN substr(Songpath, -4, 1) = '.' THEN substr(Songpath, -3, 3) WHEN

```
substr(Songpath, -5, 1) = '.' THEN substr(Songpath, -4, 4) WHEN
substr(Songpath, -6, 1) = '.' THEN substr(Songpath, -5, 5) ELSE
substr(Songpath, -6, 6) END)) > 1)\<Album>\<Format|Statistic:Count(All)>
```

33.
    ```
    <Group|name:Playlists|Show tracks:No>\Songs in Playlist|Icon:Top level|SQL
    filter: Exists (SELECT * FROM PlaylistSongs WHERE IDSong =
    Songs.ID)\<Album>
    ```

34.
    ```
    <Group|name:Playlists|Show tracks:No>\Songs Not in Any Playlist|Icon:Top
    level|SQL filter: Not Exists (SELECT * FROM PlaylistSongs WHERE IDSong =
    Songs.ID)\<Album>
    ```

35.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with avg. track rating >=
    4 stars|Icon:Top level|Filter:Songs.IDAlbum In (SELECT IDAlbum FROM Songs
    GROUP BY IDAlbum HAVING Avg(Rating) >= 80)\<Album|Statistic:Count(All),
    Avg(Rating)>
    ```

36.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with avg. track ratings
    >= 3 and less than 4|Icon:Top level|Filter:Songs.IDAlbum In (SELECT
    IDAlbum FROM Songs GROUP BY IDAlbum HAVING Avg(Rating) Between 60 AND
    79)\<Album|Statistic:Count(All), Avg(Rating)>
    ```

37.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with avg. track ratings
    >= 2 and less than 3|Icon:Top level|Filter:Songs.IDAlbum In (SELECT
    IDAlbum FROM Songs GROUP BY IDAlbum HAVING Avg(Rating) Between 40 AND
    59)\<Album|Statistic:Count(All), Avg(Rating)>
    ```

38.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with avg. track ratings
    >= 1 and less than 2|Icon:Top level|Filter:Songs.IDAlbum In (SELECT
    IDAlbum FROM Songs GROUP BY IDAlbum HAVING Avg(Rating) Between 20 AND
    39)\<Album|Statistic:Count(All), Avg(Rating)>
    ```

39.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with avg. track ratings
    less than 1|Icon:Top level|Filter:Songs.IDAlbum In (SELECT IDAlbum FROM
    Songs GROUP BY IDAlbum HAVING Avg(Rating) Between 0 AND 19)\<Album|
    Statistic:Count(All), Avg(Rating)>
    ```

40.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with all tracks rated|
    Icon:Top level|Filter:Songs.IDAlbum In (SELECT IDAlbum FROM Songs GROUP BY
    IDAlbum HAVING Min(Rating) >= 0)\<Album|Statistic:Count(All), Avg(Rating)>
    ```

41.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with some tracks rated|
    Icon:Top level|Filter:Songs.IDAlbum In (SELECT IDAlbum FROM Songs GROUP BY
    IDAlbum HAVING Min(Rating) < 0 AND Max(Rating) >= 0)\<Album|
    Statistic:Count(All)>
    ```

42.
    ```
    <Group|Name:Album Ratings|Show tracks:No>\Albums with no tracks rated|
    Icon:Top level|Filter:Songs.IDAlbum In (SELECT IDAlbum FROM Songs GROUP BY
    IDAlbum HAVING Max(Rating) < 0)\<Album|Statistic:Count(All)>
    ```

43.
    ```
    <Group|name:Playing statistics|Show tracks:No>\Played Today|Icon:Top
    level|Filter:<Days since last played> = 0\<Album|Statistic:Min(Time since
    last played)>
    ```

44.
    ```
    <Group|name:Playing statistics|Show tracks:No>\Played This Week|Icon:Top
    level|Filter:<Weeks since last played> = 0\<Album|Statistic:Min(Days since
    last played)>
    ```

45.
    ```
    <Group|name:Playing statistics|Show tracks:No>\Played This Month|Icon:Top
    level|Filter:<Months since last played> = 0\<Album|Statistic:Min(Weeks
    since last played)>
    ```

46.
    ```
    <Group|name:Playing statistics|Show tracks:No>\Played This Year|Icon:Top
    level|Filter:<Years since last played> = 0\<Album|Statistic:Min(Months
    since last played)>
    ```

47.
    ```
    <Group|Name:Multi-item tracks|Show tracks:No>\Multi-artists|Icon:Top
    ```

`level|Filter:Songs.Artist Like '%;%'\<Artist>`

48. `<Group|Name:Multi-item tracks|Show tracks:No>\Multi-genres|Icon:Top level|Filter:Songs.Genre Like '%;%'\<Genre>`
49. `<Group|name:Problematic tracks|Show tracks:No|Child of:FilesToEdit|Position:Last child>\Tracks without Lyrics|Icon:Top level|Filter:<Lyrics> = ''\<Title|Trim:1>\<Title>`
50. `<Group|name:Problematic tracks|Show tracks:No|Child of:FilesToEdit|Position:Last child>\Tracks without Comment|Icon:Top level|Filter:<Comment> = ''\<Title|Trim:1>\<Title>`

Mask storage and node order

The masks are stored in your personal `MediaMonkey.ini` file normally located in your `Local Settings Application Data\MediaMonkey` folder. In Windows XP this can be found under `x:\Documents and Settings\<your account>\` where `x:` is the hard drive containing your account files. The section to look for is `[CustomNodeMasks]`. They are named Mask1, Mask2, … Masks are numbered in the order they are created. Edit the file for later re-sorting.

Nodes under the 'Magic Nodes' root node are ordered according to their mask number.

Nodes under the standard library root node behave in the reverse fashion.

If you get a situation where Magic Nodes won't start due to errors in mask definitions then you may need to edit this file to remove or correct the offending mask.

[End of Document]