interval, the increment added to the interval after each unsuccessful reconnection attempt and the maximum interval can all be set dynamically from the configuration menu. This menu can also disable the reconnection mechanism and/or the increasing duration.

7.2.3 The Effect of the Mouse Buttons and Keys

This section describes how a user can use the mouse buttons and the keyboard to operate the MTA console. The following tables show the effect of the command buttons that can be found on the MTA console.

Top row of buttons

Button	Effect
`Refresh/Reconnect`	Refresh the data or reconnect to the QMGR.
`quit`	Exit the console program.
`connect/disconnect`	Connect to or disconnect from the QMGR.
`control/monitor`	Change between the two modes.
`configuration`	Set various parameters.
`Qmgr control`	Perform global QMGR control (only if authorised).

Channel buttons

Button	Effect
`info`	Display information about the channel.
`++`	Select next channel
`--`	Select previous channel
If connected in authorised mode	
`disable`	Disable this channel.
`enable`	Enable this channel.
`clear delay`	Clear any delays on this channel.
`add delay`	Specify a delay. The channel will wait this time before starting again.
`downward force attempt`	Invoke this channel now if possible, clearing any delays and those messages and MTAs waiting on this channel.

MTA buttons

Button	Effect
`info`	Display information on this MTA.
`++`	Select the next MTA
`--`	Select the previous MTA
	If connected in authorised mode
`disable`	Disable this MTA.
`enable`	Enable this MTA.
`clear delay`	Clear any delays for this MTA.
`add delay`	Specify a delay before next retry.
`force attempt`	Clear any delays for this MTA, an similarly for the channel this is on.
`downward force attempt`	Clear delays for this MTA and those of any messages on this MTA.

Message Buttons

Button	Effect
`info`	Display information about this message .
`++`	Select next message
`--`	Select previous message
`show all`	By default only information on ten messages is shown. This button will show all messages on this MTA/channel.
	If connected in authorised mode
`freeze`	Stop this message being delivered.
`thaw`	Allow this message to be delivered (this is the opposite of freeze).
`clear delay`	Clear the next delivery attempt time.
`add delay`	Set a next delivery attempt time.
`force attempt`	Try and deliver the message now by clearing all delays associated with this message.

Here are the effects each button has when pressed within the MTA console's display. It is based on a three-button mouse. Different configurations are tailorable through the X defaults mechanism for other hardware.

`left hand button:` If this button is pressed while the cursor is over one of the many *command* buttons in the console, that command is invoked.

If the console is in monitor mode and the left-hand button is pressed while the cursor is over the area occupied by the display of an entity, the console toggles to control mode, focuses on that entity, and displays information about the entities directly below it. For example, if in monitor mode the left-hand button was pressed while the cursor was over the area occupied by the channel `x400out84`, then the console would toggle modes, make `x400out84` the current channel and display information about the MTAs currently on that channel.

If the console is in control mode and the left-hand button is pressed while the cursor is over the area occupied by the display of an entity, then the console focuses on that entity and displays information about the entities directly below it. For example, if in control mode the left-hand button was pressed while the cursor was over the area occupied by the channel `x40084`, then the console would make `x40084` the current channel and display information about the MTAs currently on that channel.

`middle button:` If the console is in monitor mode and the middle button is pressed while the cursor is over the area occupied by the display of an entity, the console toggles to control mode and focuses on that entity.

If the console is in control mode and the middle button is pressed while the cursor is over the area occupied by the display of an entity, the console toggles the focus on that entity (i.e., if the focus at that level is not on the entity, the focus is set on that entity. However, if the focus is already on that entity, the focus is unset).

`right button:` This button has a similar effect in both modes. If it is pressed while the cursor is over the area occupied by the display of an entity then the information displayed relating to that entity is refreshed and redisplayed.

There are various "accelerators" installed in the MTA console. These are key presses that allow the user to invoke commands that would otherwise have to be invoked by moving the cursor to a specific area on the screen and then pressing one of the mouse buttons. The accelerators installed in the console all provide alternative ways to invoke some of the commands represented by *command* buttons. Some have prerequisites for use. Table 7.1 shows the accelerators, their prerequisites, and the commands they invoke. This table has two modifier keys that must be held down together with the other keys mentioned in the accelerator for that accelerator to work – the *Meta* key and the *Control* key.

Several of the popups have various text boxes into which a user can type. At any one instant only one of these boxes can have the keyboard focus (the box with the keyboard focus gets the text the user types in). This is indicated by the text box with the focus having a border around it. All other text boxes will have no border. There are accelerators installed in the MTA console to allow the user to change the keyboard focus. The up arrow key when pressed will change the focus to the box above the one currently with the focus. The down arrow key when pressed will change the focus to the box below the one currently with the focus. The tab key has the same effect as the down arrow key. If the left-hand button on the mouse is pressed when the cursor is over one of the text boxes, that text box gets the focus.

Accelerator	Command	Prerequiste
Meta + R	Refresh	
Meta + Q	Quit	
Meta + D	Disconnect	
Meta + C	Connect	
Meta + M	Toggle Modes	
Meta + F	Configuration	
Return	Ok	A popup must be on display
Control + C	Notok	A popup must be on display
Meta + H	Toggle Heuristic	The configuration popup must be on display
Meta + A	Toggle Authorisation	The connect popup must be on display

Table 7.1: MTA Console Accelerators

7.2.4 Badness

The *badness* of an entity (channel, MTA or message) is a value used to compare the state of one entity with that of another. This comparison is used throughout the MTA console to order the displayed lists of entities, to choose which colour to display that entity in, and so on.

The badness of an entity is calculated from the sum of up to four factors.

`number`: The number of messages on an entity.

`volume`: The volume of messages on an entity.

`age`: For channel and MTA entities, this is the oldest message on the entity. For message entities, this factor is the age of the message itself.

`last`: The last successful attempt on an entity.

Each factor has a specific upperbound and these upperbounds are tailorable (this is described below). In the badness summation, the factors are expressed in terms of percentages of their respective upperbounds. So if the upper bound on the number factor is represented as *ub_number* and similarly for the other factors then the badness summation would be calculated as follows: $(number/ub\_number) * 100 + (volume/ub\_volume) * 100 + (age/ub\_age) * 100 + (last/ub\_last) * 100$, or, put more concisely, $\sum^{factors}(factor/ub\_factor) * 100$.

Tailoring of Upperbounds

When the console connects to a queue, it builds a database of tailoring entries. It can get the information for the database from any of three sources. The file specified in the connect popup, the hardwired defaults and the X defaults. Entries in the file override equivalent entries in the defaults. The file specified in the connect popup should be relative either to the current directory, a full path name, or the home directory.

An entry in this database has the form:

Tailor File Values		
type	access	Key
in	mts	mtsin
out	mts	mtsout
both	mts	mtsboth
in	mta	mtain
out	mta	mtaout
both	mta	mtaboth
in		mtsin
out		mtsout
both		mtsboth
shaper		internal
warn		internal
delete		internal
qmgrload		internal
debris		internal
timeout		internal

Table 7.2: Tailor File Entries to Console Keys.

```
<key> ":" <value>
```

The `key` is a tuple indicating which channel, or type of channel and which type of entity on that channel the tailoring information in the value relates to. The channel or channel type and the entity are separated a "`-`". The entity must be one of:

```
chan
mta
msg
```

If the tailoring information is related to a specific channel, then the channel's name must be before the `-`. For example, if "822local," "x400in84," and "p2explode" are all names of channels, then the following are valid keys that can be used in the database: "822local-chan," "x400in84-mta," and "p2explode-msg".

If the tailoring information is related to a type of channel then the type of the channel appears before the `-`. The type of a channel is constructed from the `type` of that channel given in the *tailor* file and the `access` of that channel also in the *tailor* file. Table 7.2 gives all the types of channels that can be used in keys and how they are constructed. Some valid keys of this form are "mtaboth-chan", "passive-msg", "internal-msg."

When the console searches through the database, it searches first for keys with the specific channel name in them (i.e., "channelname-entity"). If this fails, it searches for keys with the channel type in them (i.e., "channeltype-entity"). This second search will never fail as there are default entries for all possible channel types.

The `value` is a comma-separated list of expressions. These expressions are of the form `keyword<upperbound`, where keyword is one of:

```
age
vol
num
last
```

and upperbound is a number and an optional letter representing a unit. The possible units are

`h:` Hour (number * 60).

`d:` Day (number * 24 * 60).

`k:` Kilo (number * 1000).

`M:` Mega (number * 1000000).

Note that if one of the four keywords is missing from a value, the associated factor is ignored in the badness summation.

Figure 7.1 shows an extract from the MTA console's hardwired defaults.

```
#define default_mtsboth_chan \
        "mtsboth-chan:age<2h,vol<1M,num<500,last<2h"
#define default_mtsboth_mta \
        "mtsboth-mta:age<2h,vol<1M,num<500,last<2h"
#define default_mtsboth_msg \
        "mtsboth-msg:age<2h,vol<1M"

#define default_mtain_chan \
        "mtain-chan:last<2h"
#define default_mtain_mta \
        "mtain-mta:last<2h"
#define default_mtain_msg \
        "mtain-msg:last<2h"

#define default_passive_chan \
        "passive-chan:age<20,vol<1M,num<10,last<20"
#define default_passive_mta \
        "passive-mta:age<20,vol<1M,num<10,last<20"
#define default_passive_msg \
        "passive-msg:age<20,vol<1M"
```

Figure 7.1: Extract From the Console's Tailoring

7.3 LINEconsole

The *LINEconsole* provides similar functionality to the control mode of the MTA console. The *LINEconsole* does not work on top of the X-windowing system and so can be used to manage the QMGR from a terminal or from the comfort of your own home over a modem connection.

The *LINEconsole* has a three level hierarchical view of the queue and its contents. These three levels are channels, MTAs, and messages. The prompt specifies which level the user is currently studying and is constructed from the names of the entities chosen in the levels above. For example, if the user was examining the MTAs on the channel `822tous` the prompt would be `822tous.mta`.

Within a level, the user can use UNIX regular expressions (*regex* (3)) to specify the entity or entities he or she is interested in. This allows easy group control. For example to disable all the X.400 protocol channels, the user would type the command "disable x400". For each channel whose name matched the regular expression "x400", the line console would then ask the user to confirm that he or she wants to disable that channel.

Note that when moving down a level in the hierarchy, the user must specify an expression that matches a unique entity at the current level.

7.3.1 LINEconsole Commands

This section provides a brief overview of the commands the line console provides.

`set` This command is used to set one of the line console's variables, e.g., `set pager=off` or `set user=foo`. Currently the variables you can set are `pager`, `tailorfile`, `authorised` and `user`.

If no arguments are given with the set command, the current settings of the variables are printed.

`status` This command prints out the overall status of the QMGR currentlt connected to.

`connect` This command connects to a specified QMGR. The line console will prompt for the name of the QMGR and, if attempting to connect in authorised mode ("authorised=yes"), a password to connect with.

`disconnect` This command disconnects the line console from the QMGR.

`quit` This command quits from the line console.

`list` This command lists all the entities on the current level with badness greater than zero [1].

If the command is given with the argument "all," entities with zero badness are also displayed.

`info` This command displays information relating to all entities which match the given regular expression.

`refresh` This command updates the line console's database form the QMGR.

`down` This command moves the line console to the level below the current level in the hierarchy. Note that to do this a regular expression which matches a unique entity from the current level must be specified.

[1] badness is calculated and tailored as with the MTAconsole

`up` This command moves the line console to the level above the current level in the hierarchy.

`next` This command selects the next entity in the current list and displays the information relating to that entity. If no entity is currently selected, the first entity on the list is selected.

`previous` This command selects the previous entity in the current list and displays information relating to it. If no entity is currently selected, the last entity is selected.

`current` This command allows the user to select a given entity. The regular expression given must select only one entity.

The remaining commands are only available if the user connects to the QMGR in authorised mode.

`enable` Enables processing for entities which match the given expression.

`disable` Disables processing for entities which match the given expression.

`delay` Delays matching entities for a specified amount of time.

`clear` Clears any delay on matching entities.

`belowclear` Clears and enables any entities below the matching entities.

`aboveclear` Clears and enables any entities above the matching entities.

`quecontrol` Performs an overall control on the QMGR. The valid controls are `abort`, `terminate`, `restart`, `rereadQueue`, `disableSubmission`, `enableSubmission`, `disableAll`, `enableAll`, `increasemaxchans`, and `decreasemaxchans`.

Note that there is a simple redirection facility in the line console. This facility allows a user to redirect the output from the commands `list`, `info` and `status` into a file. This can be either as an append operation or as an overwrite operation. To invoke this facility follow the usual command with a redirect symbol and a file name, e.g.:

```
status > /tmp/foo
info 822 >> /tmp/foo
list >> /tmp/foo
```

The `>` symbol represents the overwrite operation, and the `>>` symbol represents the append operation.

7.4 Lconsole

The *lconsole* command is an alternative line based console. Both versions of the line console will probably not be distributed in future. This version may well be the default version if its functionality is enhanced.

The *lconsole* provides basic access to the qmgr. It allows all the operations that may be achieved with the X based console, but does not have such good facilities for displaying the data. It has a number of commands that map onto the qmgr protocol. It may be run in authorised or unauthorised mode.

There are a few command line flags:

`-Q name`: Contact the given QMGR.

`-u user`: Use the given name to authenticate. A password will be prompted for if not given.

`-p passwd`: Connect with the given password. Use with care!

`-b`: Run in batch mode, little prompting is given in this mode.

Additonal arguments on the command line, are taken as commands to execute. This allows the the possibility of shell scripts top access the queue status.

All comamnds may be abbreviated to any unique string. The *lconsole* has the following commands:

`channel`: This command lists out all the channels with their number of messages and reports by default. Normal channels are printed as `channel:msgs+reports`. Active channels have the : replaced by a `*`. Channels that are disabled have the : replaced by `!`. It an argument is given, then this is taken as a regular expression to match a channel. Any channel matched has its full details printed.

`clear`: This command clears a delay. The second argument is one of the following:

- `channel`: The next argument is a channel from which to remove the delay.
- `mta`: The next two arguments are a channel and an mta from which the delay is removed.
- `message`: The next argument is a queue ID, followed by a list of recipients numbers. For each recipient of the message, the delay is removed.

`close`: Close the connection to the QMGR.

`decrease`: Decrease the number of runnable channels.

`delay`: Set a specified delay for a given channel. The second argument is one of:

- `channel`: The next argument is a channel followed by a delay given in the same syntax as the `MTAconsole`.
- `mta`: The next two arguments are a channel and an mta follwed again by a delay specification.

`message:` The next argument is a queue ID, followed by a a delay time and then a list of recipients numbers. For each recipient of the message, the delay is set.

`disable:` This command disables a channel, mta or message. The syntax is identical to the `clear` command. Additionally though, two other keys are recognised. `disable all` will disable all channels. `disable submission` will disable submission requests.

`enable:` This command is identical in syntax to `disable`, but with oppostire effect.

`help:` With no arguments, this lists out a list of valid commands. With an argument, a one line summary of the specified command is given. An argument of `?` gives a one line summary of all commands.

`increase:` Increase the number of runnable channels.

`msg:` Give details about a messages queued on a channel/mta pair. The first argument is the channel (which may be a regular expression), the second an MTA. An optional third argument will print out full details on message queue id's matching that regular expression.

`mta:` Display information about mtas associated with a given channel. An optional second argument is a regular expression which allows more detailed information about matching mta names.

`open:` Make a connection to a QMGR. The optional second argument is the user to bind as. A password will be prompted for if a user is given.

`quit:` Close any outstanding connections, and exit the process.

`rereadqueue:` Forces the QMGR to initiate a re-read of the queue on disk.

`restart:` Restarts the QMGR.

`show:` This allows messages matching certain criteria to be displayed. The expression is of type key=value. The key may be any unique abbreviation of the following: