NAME

mapper – GMA battle grid map

SYNOPSIS

gma mapper −− −−help

gma mapper [**−display** *name*] [**−geometry** *value*] [*other wish options*] **−−** [**−A**] [**−a**] [**−B**] [**−b** *pct*] [**−C** *file*] [**−c** [*image***=**]*name*[**:***color*]] [**−D**] [**−d**] [**−G** *n*[**+***x*[**:***y*]]] [**−g** *n*[**+***x*[**:***y*]]] [**−h** *hostname*] [**−k**] [**−l**] [**−M** *module*] [**−n**] [**−P** *pass*] [**−p** *port*] [**−s** *file*] [**−t** *transcript*] [**−u** *name*] [**−X** *proxyhost*] [**−x** *proxyurl*] [**−−button−size small|medium|large**] [**−−curl−path** *path*] [**−−curl−url−base** *url*] [**−−dark**] [**−−mkdir−path** *path*] [**−−nc−path** *path*] [**−−no−blur−all**] [**−−scp−dest** *path*] [**−−scp−path** *path*] [**−−scp−server** *hostname*] [**−−ssh−path** *path*] [**−−update−url** *url*] [*mapfiles...*]

gma mapper [**−display** *name*] [**−geometry** *value*] [*other wish options*] **−−** [**−−animate**] [**−−no−animate**] [**−−blur−all**] [**−−no−blur−all**] [**−−blur−hp** *pct*] [**−−button−size small|medium|large**] [**−−config** *file*] [**−−character** [*image***=**]*name*[**:***color*]] [**−−chat−history** *n*] [**−−curl−path** *path*] [**−−curl−url−base** *url*] [**−−dark**] [**−−debug**] [**−−generate−config** *path*] [**−−generate−style−config** *path*] [**−−guide** *n*[**+***x*[**:***y*]]] [**−−host** *hostname*] [**−−keep−tools**] [**−−major** *n*[**+***x*[**:***y*]]] [**−−mkdir−path** *path*] [**−−module** *module*] [**−−nc−path** *path*] [**−−no−chat**] [**−−password** *pass*] [**−−port** *port*] [**−−preload**] [**−−proxy−host** *proxyhost*] [**−−proxy−url** *proxyurl*] [**−−scp−dest** *path*] [**−−scp−path** *path*] [**−−scp−server** *hostname*] [**−−ssh−path** *path*] [**−−style** *file*] [**−−transcript** *filename*] [**−−update−url** *url*] [**−−username** *name*] [*mapfiles...*]

Options cannot be combined into a single CLI argument. They are evaluated in the order they are received. While not recommended (nor guaranteed to always be the case), it happens that options specified after *mapfile*s will affect only *mapfile*s named following those later options. Thus, if a *mapfile* has a name beginning with a hyphen, it is necessary to specify it in some sneaky way, such as prefixing it with a "**./**".

(Depending on your system, you may need to run as **wish mapper.tcl** ...)

DESCRIPTION

The **mapper** program displays a battle grid map for the players to see their tactical positions with respect to their opponents and features of their environment (e.g., dungeon rooms).

If one or more map file names are listed on the command line, those files are loaded into the grid display. Otherwise, a blank map grid is shown. Map files may subsequently be loaded via interactive commands or upon request from a control service (see the **−h** option and "Control Protocol" section, below).

See notes at the bottom of this manpage for installation requirements.

HELPER PROGRAMS

The **mapper** client makes use of a few other programs to facilitate the efficient handling of data transfers. These are only used if the client is working in networked mode rather than as a stand-alone map drawing program.

curl Assuming the GM (or whomever is administrating the map assets) has arranged for a web server to host content for the map, the **mapper** client will invoke the **curl**(1) program to actually retrieve the files from that server. If needed, you will need to provide options and/or configuration file parameters as documented below to let **mapper** know where to find the **curl** binary and what options it needs to find the web server, navigate through proxy servers, etc.

scp For users with privileges to *store* data on the web server (generally, this would be the GM or other designated administrative users, not the general player group), **mapper** will invoke **scp**(1) to copy new data files up to the server. For example, in store-and-forward mode, when opening a local data file or pushing map contents out to other clients, **mapper** checks to see if the server has the data file(s) available for everyone to download. If not, it will use **scp** to upload them to the server before sending out a command to peer clients to retrieve them.

This requires that the user is authorized to perform this action on the remote server. Usually this is done by having an SSH certificate loaded via **ssh-agent**(1) and the server set up to recognize that for the needed operations without requiring a password to be manually entered.

ssh Along with the use of **scp** to transfer new data to the web server, **mapper** will use **ssh**(1) to create directories and set file permissions as needed on the server for the content being uploaded.

OPTIONS

The following options control the behavior of **mapper**.

−A,**−−no−animate**

Suppress the animation effects enabled by the **−a** (**−−animate**) option. This is the default, so it only needs to be given to cancel an **−a** option which appeared previously or which was read from a configuration file.

−a,**−−animate** This option causes the mapper to make a lame attempt to "animate" the appearance of objects being drawn onto the screen by updating the display after each one. (The author was feeling oddly nostalgic about working with the Tektronix storage-screen computer displays of his childhood at the time. If you don't know what that means, go watch an episode of the 1970s-era Battlestar Galactica and look at most of their computer screens.)

−B,**−−blur−all** Normally, the imprecision introduced to health bar displays by the **−b** (**−−blur−hp**) option applies only to "monsters" (opponents). With this option, it applies to all participants.

−−no−blur−all Cancels the effect of the **−B** (**−−blur−all**) option.

−b,**−−blur−hp** *pct*

This option "blurs" the health bar displays by rounding off the displayed hit point total to only be accurate in *pct*-percent intervals. For example, a setting of **10** means the health bar will blur the value by 10%; in other words, rather than every hit point showing proportionally on the health bar, the health bar will only show 10 possible intermediate values, corresponding to the hit points being 1–9%, 10–19%, 20–29%, ..., 90–99% of the total, as well as 0% and 100%. Thus, higher *pct* values indicate less accurate displays.

Setting *pct* to 0 (or less) indicates that no blurring is desired; in this case the display is precisely accurate. This is the default, but note that the hit points reported may be blurred on the server (GM)'s side independently.

Once a creature reaches 0 hit points, no further blurring is done, so the bleed-out sequence is accurate (but this is fairly quick for almost all creatures and is of less consequence than the hit point totals while they are still alive and fighting, so this was considered a better course of action).

−−button−size *size*

Change the size of the toolbar buttons. The *size* value may be any string starting with **s**, **m**, or **l**, representing small, medium, or large-size icons. Small buttons are the default.

−C,**−−config** *file* Read a set of command-line options from the named *file* as if they appeared at this point in the list of command-line options. Only the long-form option names are allowed and are given without the leading hyphens. The file must contain a single option per line. Options which take a parameter are separated from their parameter with an equals sign (although this is currently not supported in the command line itself). For example, a configuration file might contain the following:

```
# My configuration settings
scp-dest=/usr/local/game-support
scp-server=www.example.org
curl-url-base=https://www.example.org/game
no-animate
keep-tools
```

Note that any line whose very first character is a pound sign ("**#**")) is ignored as a comment.

If the file **~/.gma/mapper/mapper.conf** exists, it is read first before command-line options or (other) configuration files are loaded.

Note that more than one **−−config** (and/or **−C**) option may be given, in which case the files are read in the order they appear in the command line. This may be used to split up options into different files, such as general settings common to all sessions, and specific settings based on networks or different games.

−c,**−−character** [*image***=**]*name*[**:***color*]
Add player *name* to the list of standard players tracked by the mapper. This is the list that appears in the pop-up menu for placing people onto the map. If *color* is also specified, that color is used for the threat zone highlighting. This may be a standard X11 color name, or an RGB value in the form **#***rgb,* **#***rrggbb,* or **#***rrrgggbbb.* The default is "blue".

If an image file will be used with the character that's not the same name as the character, specify it as *image***=***name* in this option.

Multiple **−c** options may be given. Each adds another name to the list.

Note that when the mapper is networked with the GM's console, the default list of names actually comes from the GM's console so it should not be necessary to specify these names to the client from the command line or configuration file.

−−chat−history *n* Limits the retained chat history to *n* messages. When **mapper** starts, it reloads the chat history it had cached from the previous session on the current *host* and *port* but if that results in more than *n* messages being in the history, the list of messages is truncated to the most recent *n* (both in-memory and in the cache file). Any additional messages received will be kept, so the actual history will be a little larger than *n* until the next time **mapper** is started. If *n* is less than or equal to 0, then no limit is placed on the history size. The default limit is 512.

−−curl−path *path*
Specify the path to the **curl**(1) program on your system, if the built-in default doesn't work for you. This is used when fetching image and map files from the server.

−−curl−url−base *url*
Specify the base URL on the data server. The files downloaded by mapper clients will be in a directory hierarchy appended to this string.

−D,**−−debug** Increase debugging output level. Multiple **−D** options further increase verbosity of debugging messages. These are displayed in a separate window.

−d,**−−dark** This option changes the default color palette to use a darker background which may be easier to look at for longer periods of time. On macOS systems running up-to-date versions of Tcl/Tk (not the default legacy version supplied by Apple), dark mode is automatically selected if the macOS session is also configured for dark mode.

−G,**−−major** *n* Make every *n*th gridline green (very thick lines). This is for major guide lines on the map.

−G,**−−major** *n***+***o* As above, but offset the major guide lines to the right and down by *o* lines. The **+** character is required, but the value of *o* may be negative, so the option "**−G** *n***+−3**" would move the lines to the left and up by 3 lines.

−G,**−−major** *n***+***x***:***y*
If expressed this way, rather than use the same offset in both directions, move the guide lines *x* lines to the right and *y* lines down.

−g,**−−guide** *n* Make every *n*th gridline red (thick lines). This is for minor guide lines. The value of *n* may be specified in all the same ways as for the **−G/−−major** option (see above).

−−generate−config *path*
Append a set of example settings to the file named in *path*. This will include the set of options you might want to configure in the mapper, most of which will be commented out. This is intended to make it easier for you to create a configuration file for the mapper without having to remember what all of the options are. In reality you won't likely need most of them.

After writing to this file, the mapper client will exit.

−−generate−style−config *path*
Append a set of example style settings to the file named in *path*. This will include all possible style definitions. Their built-in default values are also provided, so that if this file were loaded as the *style.conf* file of the mapper, it should result in the same appearance it normally would have. The intent is to help you get started configuring the styles of the mapper and to see what the mapper's settings would be if you didn't override them. You should delete or comment out any entries you wish to leave at the built-in default values.

After writing to this file, the mapper client will exit.

−−help Print a summary of the command invocation options and exit.

−h,**−−host** *hostname*
Connect to a map control service running on the designated host. This will send updates to item positions, display of rooms, etc. If this option is not specified, no control connection is made, and the mapper runs in stand-alone mode.

−k,**−−keep−tools** Normally, map clients have their toolbars turned off to maximize the available screen space for the battle map. The GM can turn on and off their toolbars from his console as needed. If this option is given, this causes the client to unconditionally display its toolbar anyway. This is used for the main map run by the GM or whomever else is managing the group map and needs the toolbar active, or if people just want to keep the toolbar all the time.

−l,**−−preload** Pre-load all the cached images into memory at start-up, instead of loading them as needed during the map's operation. Note that this only loads cached images which are new enough that the mapper wouldn't check the server for newer versions anyway, thus allowing a mapper client to be restarted mid-game with a minimum of impact to game performance.

−−mkdir−path *path*
Specify the *server-side* path to the **mkdir**(1) program which will be used when uploading files *to* the data server (authorized users only).

−M,**−−module** *module*
Use the module ID code *module* for this session. This is used to differentiate server-side resources between campaigns which have conflicting names. This is only needed for the mapper clients used as the forwarder in store-and-forward mode (typically the GM's own client).

−−nc−path *path* Specify the path to the **nc**(1) program which will be used when sending files *to* the data server (authorized users only) through a SOCKS proxy server.

−n,**−−no−chat** Suppress the display of incoming chat messages including die rolls.

−P,**−−password** *pass*
For servers which require authentication, this specifies the password to gain entry to that server. This is a fairly simple authentication mechanism intended to block nuisance connections, spam, and accidental connections of legitimate clients to the wrong game server. If *pass* is given as a single question mark ("**?**"), then the user will be prompted to enter their password manually when connecting to the server. This avoids placing the plaintext password on the command line or in a configuration file.

−p,**−−port** *port* If the **−h** (**−−host**) option is given, connect to the specified TCP *port* number on that host. The default is port 2323.

−−scp−dest *path* Specify the *server-side* directory into which files will be uploaded (authorized users only). This will be the top-level data directory for the mapper; subdirectory names will be appended to this string.

−−scp−path *path* Specify the path to the **scp**(1) program which will be used to send files *to* the data server. (Authorized users only.)

−−scp−server *hostname*
The host name of the storage server. Only used when sending files *to* the server (authorized users only).

−−ssh−path *path* Specify the path to the **ssh**(1) program used to send files *to* the storage server (authorized users only).

−−style *file* Loads custom style definitions for fonts, colors, and so forth from the named *file*. If this option is not specified but the file **~/.gma/mapper/style.conf** exists, that file will be read by default. See **style.conf**(5) for details about what can go in this file.

−t,**−−transcript** *path*
Records all chat window activity (including results of die rolls) to the specified file *path*. If this file exists, it will be appended to with the new information.

The following special tokens may appear in the *path* string, which will be replaced with values based on the time of day the file is opened:

%a Mon, Tue, etc.

%A
Monday, Tuesday, etc.

%b
Jan, Feb, etc.

%B
January, February, etc.

%d
Day of month (1−31).

%j Julian day of the year.

%m
Month (01−12).

%y Year (00−99).

%Y
Year (all digits).

%H
Hour (00−23).

%I Hour (01−12).

%M
Minutes (00−59).

%S
Seconds (00−59).

%p
AM or PM.

%D
Date (%m/%y/%d).

%r Time (%I:%M:%S %p).

%R
Time (%H:%M).

%T
Time (%H:%M:%S).

%Z
Time zone name.

−−update−url *url* Specifies the URL where updated versions of the **mapper** program may be obtained. This enables automatic upgrades. The **mapper** will, with the user's approval, download updated versions of itself from this URL and install them.

−u,**−−username** *name*
Sets the name used to identify you amongst the other players on your server. If this option is not provided, your current system username will be used instead.

−X,**−−proxy−host** *host*[**:***port*]
For sending files *to* the server (for authorized users only), use the specified SOCKS5 proxy server. (E.g., **−X proxy.example.org:1080**.)

−x,**−−proxy−url** *proxyurl*
Use the given URL to connect through a proxy server to fetch image data. This does not affect the connection to the map server used by GMA (yet). (E.g., **−x http://proxy.example.org:1080**.)

INVOCATION

As of this writing, the mapper still has not been ported to the new GMA code in Python, and is still implemented as a Tcl/Tk script. This means you need to have a Tcl/Tk interpreter installed on your system. (See **http://tcl.tk** if you need more information about that.) Since it's a GUI application, it is run using the **wish** command (the Tcl Windowing Shell).

We have noted that at least on the Mac platform, the **wish** program will refuse to let you expand the map window larger than the largest dimensions of the screen(s) when it was launched, so you want to plug in any projector or external displays before starting the map.

INTERACTIVE USAGE

The mapper shows the dungeon area around the players and includes features which are helpful for managing game mechanics, particularly those relating to combat. It is intended to be fairly self-explanatory (and I don't have time to thoroughly document everything at the moment), so the following brief notes will hopefully suffice to help a new user navigate its quirks.

The system menu bar is not used for this application, and is left to whatever the **wish** program sets it to for generic scripts. Instead, all of the interaction with the mapper is done through its toolbar and context menu.

Tool Bar

Across the top of the map is a graphical toolbar. Click on each button to activate its features. Note that some of these turn on/off different modes of operation for the map. When this happens, the mouse cursor will change to show the mode the map is currently in.

Each button is described briefly below. The first block of buttons control the mapper's mode of operation. They function as radio buttons (only one is active at a time, and selecting one de-selects the previously selected one).

Line Tool (cross-hair cursor)
Selects the line drawing tool. When this tool is active, click the left button to start drawing a line, then click it again at the other end of the line. You may continue clicking to get multiple connected line segments (which all count as a single object on the map). When finished, press

the Escape key or click the middle button. Cancel by pressing Escape or the middle button without having defined any points on the line at all. Note that the current FILL color (not OUTLINE color) is used to draw the line on the map.

Rectangle Tool (square cursor)
: Selects the rectangle drawing tool. When this tool is active, click the left button where you want one corner of the rectangle to go, then click again where the diagonally opposite corner should go. The rectangle will be outlined in the OUTLINE color and filled in with the FILL color. Cancel by pressing Escape or the middle button.

Polygon Tool (polygon cursor)
: This works like the Line Tool except that the region inside the shape defined by the line segments is filled in with the FILL color, while the outline is colored in the OUTLINE color.

  Note that when this tool is selected, the two option buttons become active, offering some different options for how the lines of the polygon are to be joined:

  Corner
  : Each time you click on this button, it cycles through the different corner-join options: beveled, mitered, and round.

  Spline
  : Each time you click on this button, it cycles through the spline levels from 0 (no splines, just straight lines), to 9 (use 9 lines between points to make a smooth curve).

Ellipse Tool (circle cursor)
: This works like the Rectangle Tool except that it draws an elliptical shape inscribed within (tangent to) the rectangular area defined by the two mouse clicks.

Arc Tool (diamond crosshair cursor)
: This tool is for making various semicircular shapes. Its operation is a little more complex than the others. When this tool is active, the option is also active, allowing you to choose the type of arc to create:

  Arc type
  : Each time you click on this option button, it cycles through the choices of arc types: pie slice, chord, and arc.

  First, draw the elliptical shape for the arc (as if it were a complete ellipse) as described for the Arc Tool. Then, move the mouse horizontally to rotate the arc and vertically to adjust the length of the arc. When satisfied, click the left button to complete the arc. Cancel by pressing Escape.

Text Tool (i-beam cursor)
: This is used for placing text on the map. Its operation works much like the stamp tool (q.v.), in that left-clicking on the canvas will place a new copy of the current string at that location. If there is no current string, you will be prompted to enter one. Right-clicking will prompt you for a new string rather than using the current one. The current string is displayed below the tool bar.

  With this tool active, a font selection button is available. Clicking this toggles the font selection dialog. Changing the font in that dialog will alter the font of the most recently placed text object (as long as the text tool remains active) and sets the font for future text objects.

  There is also an anchor selection button while this tool is active. This shows as a centered cross (+) to indicate that the text will be centered around the point where the mouse is clicked. Clicking on the anchor selection button will cycle through all of the anchor directions available: north, south, northeast, etc. These mean that the text will be aligned so that the point where the mouse is clicked will be that direction from the text. Thus, for example, selecting an anchor of "west" (incidated by a left-pointing arrow) will center the text vertically but align it horizontally so that the point is to the left of the text.

Move Tool (iron cross cursor)
: This is the default mode, and the one you should keep the mapper in when not changing the map features. With this mode, you can drag creatures around the map as described below.

If the mouse is not over a creature token when starting to drag the mouse, the map grid itself is dragged, providing an easy way to scroll the map.

If you hold down the shift key while clicking the left button on the canvas in this mode, it will briefly show a marker to draw attention to the grid square the mouse is in.

Cut Tool (skull cursor)
: With this tool active, any object you click on with the left button will be deleted from the map immediately (no saving throw). if you click where there are multiple overlapping objects, you will be prompted to select which to delete. Press Escape if you don't want to delete any of them.

Object Move Tool (multi-arrow cursor)
: This tool allows the map objects (as opposed to creature tokens) to be dragged to new locations. Note that you are dragging the object's *reference point* with the cursor. Once an object has been moved any distance with the mouse, the arrow keys (or the standard **vi**(1) movement keys) may be used to "nudge" the object by one pixel at a time up, down, left, or right; additionally the keys **u**, **d**, **f**, and **b** may be used to move the object up, down, to the front, and to the back in the stacking order (z coordinate), respectively.

Stamp Tool (star cursor)
: This allows graphical tiles to be "stamped" onto the map. If there is a current tile already chosen, a new copy of it is placed on the map with the upper-left corner at the point the mouse was clicked. If no such tile was chosen, you will be prompted for its name. Right-clicking will force the selection of a new tile image rather than re-stamping the current one. See **rendersizes**(6) for more information about the format of these tile files. They should be rendered and (if using a map server) uploaded ahead of time so they are visible in the map.

The next block of buttons control the appearance of any new objects added to the map.

Fill Mode
: Clicking on this button toggles whether the shape will be filled or not. (Somewhat counter-intuitively, lines are filled with the FILL color, not the OUTLINE color, so turning off fill will just give you invisible lines.)

Fill Color
: Clicking on this button selects the FILL color to be used to fill in new object areas. This is disabled if fill mode is turned off.

Outline Color
: Clicking on this button selects the OUTLINE color to be used to draw around new object areas.

Grid Snap
: Clicking on this button cycles through the grid snap options:

  Off — Points may be added anywhere on the canvas (free form drawing).

  1 — Points may only be added at the intersections of grid lines.

  1/2 — Points may be added at grid intersections, and 1/2 way between them horizontally or vertically.

  1/3 — Points may be added at grid intersections, and every 1/3 of the way between them horizontally or vertically.

  1/4 — Points may be added at grid intersections, and every 1/4 of the way between them horizontally or vertically.

Line Width
: Clicking on this button cycles through the line widths from thinnest to thickest.

The next block of buttons clear the map:

Clear Features
: Clicking this button wipes the map clean except for creatures.

Clear Creatures
: Clicking this button removes all creatures from the map.

The next block gives access to tactical displays.

Toggle Combat Mode
: Normally, the GM console will automatically turn on combat mode, but if you want to manually enable or disable it, click this button. When active, the threat zones around each creature are highlighted using colored cross-hatch patterns.

  If health tracking is in effect (i.e., for creature objects which have a non-empty **HEALTH** attribute), a health bar is displayed across the bottom of each creature's token. The appearance of this bar depends on the current health of the creature. For the description that follows, the significant health statistics are:

  t The total number of hit points the creature has when at maximum health.

  x The extra points (below zero) which define the amount of lethal damage a dying creature can sustain before being dead. In Pathfinder and compatible d20 games (and perhaps others), this is the Constitution score for the creature.

  l The number of hit points worth of *lethal* damage sustained by the creature.

  n The number of hit points worth of *non-lethal* (i.e., subdual) damage sustained by the creature.

  The health bar indicates graphically the creature's health condition and relative amount of damage they have taken, as follows:

  Full health A creature in full health will have a solid green bar across the entire width of their token's space on the map.

  Injured The full width of the token space represents the creature's total (maximum) hit points (t). A red bar will start encroaching over the green in proportion to the number of lethal hit points (l) they have taken. A yellow bar will likewise represent the number of non-lethal hit points (n) taken. Thus, the health bar will be shifting more from green to red/yellow as the creature gets more and more injured, until as it nears the point of meeting its maker, the entire bar will be red.

  Flat-footed A flat-footed creature (which does not also have any of the conditions listed below) will have a blue frame around the health bar.

  Staggered When staggered due to non-lethal damage (i.e., $n>0$ and $l+n=t$), the health bar has a yellow frame around it. The creature will move to unconscious if it suffers more damage.

  Unconscious When unconscious due to non-lethal damage (i.e., $n>0$ and $l+n>t$), the health bar has a violet frame around it.

  Disabled When disabled, a red frame will appear around the health bar. The mapper will automatically assume disabled condition if a creature has exactly 0 hit points left (i.e., $l=t$.)

  Dying When at negative hit points but still above the death level ($-x<t-l<0$), a red frame will appear but the red bar will retreat to the left as more lethal damage is taken, until it's fully black at the point of death.

  Stable If dying but stabilized, the health bar will have a brown frame around it.

  Dead When completely mortally wounded ($t-l<=-x$), the health bar is solid black.

Show HP Values

This toggles the display of health statistics for players (not monsters) over the health bars. If only lethal damage has been inflicted, it displays "*hp*/*max*" where *hp* is the current number of hit points remaining, out of a maximum of *max* hit points. If non-lethal damage has been suffered, then the display is "*hp*(*nl*)" where *nl* is the amount of non-lethal damage. If a creature is fully dead, it simply says "**DEAD**".

Spell Area of Effect

This adds a spell area of effect to the battle grid. Once created, this becomes a permanent map feature which may be removed using the Cut Tool (q.v.). When this tool is activated, two option buttons are enabled which allow you to control the shape of the spell area:

Shape This button cycles through the supported spell shapes: radius, cone, and ray.

Spread This button toggles whether the spell effect "spreads" around corners. This is not yet implemented.

Select the point of origin for the spell by clicking the left button over a grid intersection (the tool will snap to intersections automatically). Then move the mouse to the target point of the spell and click again to complete the area. As you move the mouse, the spell's area and affected grids will be shown. The area of effect is filled in with cross-hatch patterns in the FILL color. Cancel by pressing Escape. This is an active tool like the other drawing tools. When finished, select another mode such as the Move Tool.

Ruler Tool

Selecting this tool allows you to measure the distance along a path. Click the left button on a point, then move the mouse to another point. If desired, multiple points may be clicked to build a path. Middle-click or press Escape to end the measurement.

Grid Display Toggle

Clicking this button turns on and off the display of the gridlines on the map. This is a local display setting only, and is not broadcast to other clients.

Die Roller

If connected to a server, this button brings up the chat window. In this window, you may send and receive messages and die rolls to other connected users. This window is split into three adjustable panes, described individually below. The division between each pane may be moved by dragging the mouse over the separation point or pane handle.

Chat Messages Pane

In the main portion of this pane displays incoming chat messages. Each is prefixed with the name of the sender. If the message was addressed only to you, the tag "*(private)*" is added after the sender's name. If it was sent to a specific subset of users, their names will be listed as "*(private to alice, bob, charlie)*".

There are two entry lines below the chat window. The top one is for sending chat messages. Anything typed in the entry box will be transmitted when the Return key is hit. To the left of this entry box is a menu button which controls who the message is sent to. If "(all)" is selected, the message is sent to all listening clients (which need not be listed in the menu; the message will be sent to everyone at the server level). If a recipient's name is selected, it will only be sent to them. If another recipient's name is selected, they are *added to* the list of recipients. These selections are actually toggles—selecting a recipient's name again will remove them from the list. This allows for messages to be sent to any arbitrary subset of users. Selecting "(all)" will clear all selections again. The "refresh" button to the right of the entry box will update the recipient selection menu with the current set of logged-in users.

The bottom entry line is for making die rolls. Into the entry box you may type any die roll string such as would be accepted to the **DieRoller.do_roll()** method as documented in **dice**(3). When the Return key is pressed, this die roll is sent to the server, which will roll

the dice and transmit the results just like a chat message (which includes the currently-selected chat recipient list). The "(i)" button to the right of the entry box will bring up a help window explaining what may be entered for die rolls.

Recent Rolls Pane

The most recent 10 rolls entered into the above-mentioned entry box are kept in a list in this pane, with the most recent on top. Clicking on the die button next to any of these will re-roll it again. If additional modifiers are in play, they can be typed into the entry box next to the die button. Whatever is entered is simply appended to the original die expression after a plus sign. Thus, entering "5" will add "+5" to the roll, and entering "1d6 fire+3" will add "+1d6 fire+3" to the roll.

Preset Rolls Pane

A set of commonly-needed die rolls may be pre-set into the tool and then invoked using the third pane. Clicking the "(+)" button will add a new preset by prompting for its name, description, and die roll. The name uniquely identifies the preset within the list. The description will appear as a tooltip for your reference when looking at your presets. The new preset is saved on the server and will be loaded into your client every time it's started. Presets are invoked in the same manner as described above for recent rolls. Clicking the "(−)" button removes the preset from the list.

If a preset name includes a vertical bar (e.g., "**12|WillSave**"), then only the part after the bar will be displayed on-screen, but the entire name is used to sort the presets in the window. This allows arbitrary sort ordering without cluttering the display.

The file load and save buttons at the bottom of the pane are used to load and save the preset list to local disk files which have the format documented in **dice**(5).

The final block of tool buttons control global operations of the mapper:

Zoom In
: Double the visual size of grid blocks.

Zoom Out
: Halve the visual size of grid blocks.

Un-Zoom
: Restore the zoom level to normal.

Load
: Add all the map objects from a disk file onto the map tool, replacing all the map features previously on the map (but not the creatures).

Merge
: Like Load, but add to the existing objects rather than replacing them.

Unload
: All the objects saved to a selected disk file are *erased* from the map.

Push
: Push the entire contents of this map client to all other clients, replacing their current contents. (Only available in store-and-forward mode, generally only for GM use. Since the server now tracks game state and clients and re-sync with it directly, there is no longer a need for clients to push their contents to each other, and that was a problematic operation anyway.)

Store and Forward
: Toggles store-and-forward mode. When enabled, this changes the behavior of the following other buttons, providing a client update path that is much more efficient and less error-prone than streaming the object updates through the server. Stand-alone (non-networked) map clients should use the normal mode of operation instead.

  Load
  : Prompts for the selection of a map file from disk as usual. However, rather than loading that file directly, it checks to see that the file is available from the server by checking the local cache and (if necessary) downloading from the server. If the file is not found by those operations, it will be uploaded to the server (assuming the user has the proper SSH access active at the time). Other clients are then instructed to load the map file from the server.

Merge As with the Load button, but merges the map file with the existing map contents rather than replacing them.

Unload Ensures that a server copy of the map file exists as the Load button does, but then instructs the remote clients to delete the contents of that file rather than sending individual object deletion commands over the network to them.

Push Saves the current map contents to a temporary file, uploads it to the server, and then instructs the other clients to load that file.

Sync *(Note that this button's function has changed as of version 3.25.)* This clears the contents of the mapper client and requests a fresh set of data from the server, thus synchronizing this client to be in line with the server's idea of the current game state. Depending on how the server is configured, it may automatically perform this operation for you when you connect to it.

Save Save everything on the map to a disk file. You will be prompted to decide whether this includes creatures as well.

Exit Exit the mapper program.

Context Menu

Clicking the right button over an object calls up a context-sensitive menu with the following options. Not all options will be enabled in all cases. Most of these involve performing actions on creatures.

Remove *name* Remove the creature from the map. If there are multiple creatures in the same grid, a submenu will allow you to select which one to remove, or allow you to remove them all.

Add Player... Add one or more new player tokens into the grid clicked. This pops up a dialog box to enter the relevant information about the new player:

name The name by which the creature is to be known on the map. This *must* match the name the GM console is using to track initiative, or it'll never be highlighted when its turn comes up (otherwise the name doesn't matter). If the name coincides with graphical tile images already loaded, that image will be used instead of a plain circle with the creature's name inside. If a range in the form **#***n***−***m* is appended to the name (usually with a space between the name and this notation), then *m*−*n*+1 copies of the creature are added in a series of grid spaces starting with the one right-clicked and continuing to the right. For example, entering the name "Orc #1−3" will create three creatures, named "Orc #1", "Orc #2", and "Orc #3". Names must be unique. If another token was already on the map with the same name, it is replaced with this one.

If a different image file is needed than the default (named the same as the person's name), specify it as *imagename*=*creaturename* (optionally followed by the **#** notation described above).

size The size, in units of grid squares (diameter), of the creature's token. You can also use standard size designations **f** (fine), **d** (diminutive), **t** (tiny), **s** (small), **m** (medium), **l** (large), **h** (huge), **g** (gargantuan), **c** (colossal). Where it makes a difference, indicate "tall" creatures by using a capital letter and "wide" creatures with a lower-case one. Since the recommended practice is to use the size codes, which means you would use the same code for both *size* and *area* fields, any time you type into the *size* field, that will update *area* at the same time. If a different *area* is needed, that can be edited afterward separately.

area The threat area in the same units as the *size* field. This may also be one of the standard size designator codes as with *size* (and this is generally preferred). In that case, for size categories larger than medium, use upper-case (tall) letters for size categories of tall creatures, and lower-case for

long creatures.

color The color of the threat zone to draw around the creature in combat mode.

reach? Check this box if the creature has a reach weapon in hand.

Clicking **Ok** places the creature(s) on the grid and dismisses the dialog box, while clicking **Apply** places the creature(s) but leaves the dialog up in case you want to add more creatures to that grid square.

Add Monster... Just like **Add Player** but adds a monster token.

Toggle Death for *name*

Flips the creature token between living and dead states. The mapper will automatically draw an "X" across the creature token in addition to switching to the "dead" image (if images are used).

Toggle Reach for *name*

Flips on/off the extra threat zone for reach weapons.

Toggle Spell Area for *name*

Defines a spell effect which is described as a radius "centered on you" (or some creature). After choosing this item, click the left button to define where the radius extends from the creature's perimeter. If there was already a spell in effect, this cancels it. This differs from the spell area tool from the toolbar in that it moves with the creature and radiates from the creature's entire space rather than coming from a fixed point on the map. The area is filled in with the current FILL color.

Polymorph *name* If alternative images are available for a creature, this selects which is to be displayed. If the creature has a **SKINSIZE** attribute which indicates the size of each of its polymorphed forms, then this menu will allow you to choose between the number of forms defined for that creature, and will automatically adjust the creature size at the same time. Otherwise, the mapper program doesn't know what alternate forms are available so it will offer you a choice of three different forms, and will make a best-effort attempt to locate and display the corresponding images. In this case, you will need to manually adjust the size if needed.

Change Size of *name* Alter the size of a creature token.

Toggle Condition for *name*

Selects a condition from the list of conditions built in to **mapper** or defined by the map service for custom game-specific conditions. If the selected condition is already set for the target creature(s), then it is removed. Otherwise it is added to the target(s).

Tag *name* Add a tag to a creature token to indicate their conditions. The recent tags which were set are remembered and available in a sub-menu for convenience.

Set Elevation of *name* Specify how high above (or below) the obvious reference level a particular creature is. This puts a tag in the upper right corner of their token in which is shown their elevation (as a simple number). The sub-menu triggered by this item allows easy selection of relative distances, so you can quickly note that a creature moved up by 10 feet, for example. Any arbitrary elevation may be directly input by selecting "**(set)**" and typing the desired elevation into the dialog box that appears. If an absolute number is input, that will be the new elevation. If the number begins with a **+** or **−** sign, its value will be added or subtracted to the current elevation instead.

Set Movement Mode Various modes of locomotion are denoted in the elevation tag (q.v.) by using a different color for each. Use this menu item to select the mode currently employed by the creature:

land (white text on a black background)

fly (black text on a deep sky blue background)

climb (white text on a forest green background)

swim (white text on a teal background)

burrow
(white text on a sienna background)

Deselect All Cancels the multiple-creature selection.

Show Visible Objects Moves the scrollbars to bring map features into view.

Sync Others Views Moves all the other map clients scrollbars to see what this client is showing.

Refresh Display Redraws the contents of the local mapper client. This does not reload any data (see the **Sync** button in the toolbar for that), but just locally re-draws everything again. This is useful, for example, if the client didn't know about image data for tiles or creature tokens when it first rendered the display. Often, it will work in the background to discover the missing image data, so refreshing the display will then render everything properly.

name Add a player token for the named player to the map, or move it to this location if it was already on the map.

Creatures

With the Move Tool (q.v.) selected, click and drag creatures to move them around the map.

If you hold the control key down while left-clicking on creature tokens, it toggles whether that creature is included in the group selection. When a group is selected, dragging any member of the group moves the entire group at once. Context selections will also apply to the entire group (e.g., to toggle death for all the selected tokens).

In combat mode, the area threatened by each creature is shown as a dashed outline, and is cross-hatched when that player's turn is up for action. Arrows are drawn between creatures in range to be melee targets.

CONTROL PROTOCOL

When connected to a control service (see the **–h** option above), the mapper and remote service exchange commands to indicate changes to the map display. Each command is a newline-terminated line of plain ASCII text in the following formats. If the mapper sends one of these commands to the service, it is indicating that the local user made those changes and requests that other connected map clients update themselves accordingly. If the mapper receives the command from the service, it should comply to make the corresponding change to itself.

The software SHOULD allow Unicode text encoded as UTF-8 everywhere (of which 7-bit ASCII is a subset), but this has not been extensively tested, thus the above note that the file is plain text ASCII.

Each command must be a properly-formed TCL list value (in simplest terms, a space-separated list of elements with curly braces surrounding elements with embedded spaces; braces must be balanced). The first element of the list determines the request being made, and therefore the meaning of the other elements of the list.

Unless otherwise noted, there is no response expected from these commands.

This describes protocol verion 333. These notes are intended to be detailed enough to implement a client from and should be considered the definitive reference to the protocol.

See also the documentation of the map file format in **mapper**(5) which defines some of the data items referenced here.

Where boolean values are indicated, clients SHOULD use 0 for false and 1 for true, but MAY accept any non-zero value as true.

In the protocol command descriptions, **BOLD** text indicates literal text to be sent as-is, *Italics* indicate a

parameter whose value should appear in place of the name shown, and [square brackets] surround optional items which may be omitted.

// [*args*...]
: This is a server comment. The entire command should be ignored by all clients. This is used to inject informative context and messages into the client/server conversation which may be of interest for debugging or interactive use.

  It is permitted for clients to use certain comment strings to carry useful information the systems administrator wishes to communicate to them. These are strictly speaking not part of the server protocol, but this allows for some local configuration management to take place, and these are easy to add to the server's initial greeting to the clients. The following special comments are currently recognized:

  // CALENDAR // *system*
  : Declares which calendaring system the various dates and times are based upon. In the future this may be promoted to a protocol command, but for the time being the mapper doesn't necessarily need to know this so it's an advisory comment for now.

  // MAPPER UPDATE // *version file*
  : The latest available **mapper** client is *version* and is available for download from *file*. Mapper clients which support automatic upgrades will offer to download this version and install it for the user. For this to work, the mapper client must see this comment before any non-comment commands from the server. The designated file is fetched from the server location indicated by the value of the **−−update−url** option.

  // CORE UPDATE // *version* [*file*]
  : The core GMA code's current release version is indicated as *version*. If *file* is present, a new GMA release may be downloaded from that file in the same manner as for the mapper updates. Otherwise it is expected that the user obtains the code independently such as pulling from the git repository. Note that the git repository may contain more recent commits than this advertised release version. (This is intended to announce release points rather than individual commits.)

  // BEGIN *id max title*
  : Start tracking progress for some event identified by *id* which will have a maximum value of *max*. (If *max* is the special value "*" then there is no known maximum value yet.) The user-facing title for the event is given by *title*.

  // UPDATE *id value* [*newmax*]
  : Update the progress indicator for event *id* to have the new *value* toward the previously-declared maximum value. If *newmax* is specified and is not the empty string or "*", then the expected maximum value is updated as well.

  // END *id*
  : Stop tracking progress for the specified event *id*.

AC *name id color area size*
: Add a character to the pop-up menu for map clients. This makes it easy to place important character tokens on the map and ensures that such tokens are given consistent *id* values rather than generating a random one. Clients should NOT send this command to each other; it is intended for the server to send to the clients.

  name
  : The name of the creature as displayed on the map. As with the **−c** option, this has the format [*image*=]*name* which allows for creatures whose image names are different than their character name.

id The fixed ID for this creature. This will be used for internal references to this object on the map.

color The color to use when shading in the creature's threatened space when it has initiative to act. The *color* value may be in the form **#***rgb*, **#***rrggbb*, **#***rrrgggbbb*, **#***rrrrggggbbbb*, or a color name as defined by the local system. (For example, on a Unix-like system which uses the standard *rgb.txt* file, a certain shade of blue with 8-bit values for red=0x46, green=0x82, and blue=0xb4, could be specified as "**#4682b4**" or "**steel blue**").

area The size of the area threatened by the creature on the map. This can be the number of squares in diameter, but more commonly it should be a standard size code such as "**M**" for medium-size creatures.

size The size occupied by the creature on the map. This can be the number of squares in diameter, but more commonly it should be a standard size code such as "**M**" for medium-size creatures.

ACCEPT *msg_set* A client sends this message to the server to indicate that from this point forward (until another **ACCEPT** command), only the commands listed in *msg_set* should be sent to it by the server. This is a list of command names as they appear in this specification (e.g., **{AI CO TO}**), or it may be the special value "*****" which means to accept all messages (the default).

AI *name size* Add an image to the map. This begins a sequence of commands which together transmit image data to the client, followed by a number of "**AI:**" commands and finally an "**AI.**" command at the end. Other commands MAY appear between any of these commands while the image is being transferred, but another "**AI**" command is NOT allowed until the previous one is completed. This does not *draw* the image on the map; it merely defines it so the client knows what to draw when an image of the given *name* at the specified *size* is called for.

name The name of the image as it is referred to by other protocol commands and inside map files.

size The zoom factor desired. This is a real number where 1 (or 1.0) is the default zoom level for the map. If the map is zoomed out one step, images of *size* 0.5 will be needed; if zoomed in one step, *size* will be 2 or 2.0. The map client does not internally have the ability to scale images, so it uses a separate image data file for each graphic element at each desired zoom level.

Use of a web server hosting images and the protocol command "**AI@**" are preferred to sending raw image data through the mapper data socket.

AI: *data* Add a line of image data to the image started by a previous "**AI**" command. The *data* are base 64 encoded as a single unit. In other words, the binary image data (which must be in GIF format) is encoded as a single block of bytes into a single base-64 string. This string is broken into a number of pieces, which are sent in individual "**AI:**" commands. After the last "**AI:**" command is sent, finish by giving a "**AI.**" command.

AI. *lines checksum* Complete the image transfer begun by the most recent "**AI**" command.

lines The number of "**AI:**" commands which should have been received during the transfer.

checksum The SHA-256 checksum, encoded in base-64, of the original binary GIF image data before it was encoded and transmitted. This field MAY be omitted if a client is not able to calculate the checksum or chooses not to do so.

Clients SHOULD reject image transfers which fail to match the expectations given in this command as to number of data lines received and checksum.

AI? *name size*
: Request for the named image. Clients may send this if they need an image of the given *name* and *size* (as described above for the **AI** command) but no such image is defined yet in the client. This queries the server or connected peers to see if any of them know of the needed image. The client should continue to process tasks without waiting for a response (which is never guaranteed). If another peer knows of the requested image, it should respond with an "**AI**" command containing the full image data or (preferably) an "**AI@**" command indicating how to obtain the image from the game server.

AI@ *name size id*
: Like "**AI**", but rather than receiving image data directly, this merely informs the client that the image with the indicated *name* and *size* (as described above) may be obtained from the game server under the file ID *id*.

  As currently implemented, the **mapper** client checks to see if it has a cached version of the file. If so, and that file is newer than 2 days, it is used without further checks. If the cached file is older, the server is queried to see if it has a newer version of the file; if so, that is retrieved and cached; otherwise, the existing cache is updated to note that it was the known latest version as of that moment. If no usable cache file is available, the file with the given *id* is obtained from the server. For example, if the image *id* were "**abcdefg**", the URL of the file to be retrieved would be *base***/a/ab/abcdefg.gif** where *base* is the base URL as specified to the **−−curl−url−base=***base* option (or **curl−url−base=***base* configuration file line).

  Servers SHOULD be set up to provide alternative file formats so that, for example, with ID **abcdefg** files such as **a/ab/abcdefg.jpg** or **a/ab/abcdefg.png** may be retrieved for clients which use those other formats. Due to implementation limitations of the framework used, **mapper.tcl** always uses GIF format files.

ALLOW *feature-list*
: The client MAY send this command to the server after logging in. This tells the server that the client supports one or more optional features. The *feature-list* parameter is a TCL list, each element giving the name of a feature the client supports. If the list is empty, the server assumes no optional features are supported. Currently, only one feature is recognized here:

  DICE-COLOR-BOXES
  : The client supports formatting controls in the die-roll title string sent to the server in the **D** command and received from the server via the **ROLL** command. Specifically, the title may consist of multiple sub-titles separated by U+2016 characters (‖). Each of these may also have a suffix consisting of the U+2261 (≡) character followed by a color name or **#***rrggbb* color code, indicating the foreground color for that part of the title. A second such suffix may be added to indicate the background color. If this feature is not enabled, these formatting codes are stripped out by the server before sending die roll results to a client which isn't prepared to interpret them.

AUTH *response* [*user client*]
: If the server included a challenge string (see the **OK** command below), then it requires a password before the client is allowed to join the server. This is intended to prevent accidental connections by clients to the wrong servers, and to filter out nuisance connections from spammers or other random connections, so it is a fairly simple authentication mechanism using a password shared by all clients in a play group.

  If provided, the *user* parameter gives the local username, and *client* describes the client program connected to the server. These are intended to help diagnose issues

with communications between clients while they are occurring. The *user* name may be any arbitrary name except "**GM**" (unless the client is really the GM), but for best results it should be a short, single word such as the player's name or the name of the character they're playing. If the client is authenticating using the GM's credentials, the *user* parameter will be ignored, and the name "**GM**" will be used instead. Given:

C is the server's challenge as a binary string of bytes (after decoding from the base-64 string actually sent by the server),

H(*x*) is the binary SHA-256 hash digest of *x*,

P is the password needed to connect to the server, and the notation

x‖*y* means to concatenate strings *x* and *y*,

To calculate the *response* string, the client must do the following:

(1) Obtain *i* by extracting the first two bytes from *C* as an unsigned, big-endian, 16-bit integer value.

(2) Calculate *D*=H(*C*‖*P*).

(3) Repeat *i* times: Calculate *D*'=H(*P*‖*D*); then let *D*=*D*'

The binary value of *D* is then encoded using base-64 and sent as the *response* value.

Starting with protocol version 332, the server is guaranteed to respond to the **AUTH** command with either a **DENIED** or **GRANTED** message (q.v.). The client may wait for this response before proceeding with its other operations, so that it knows for sure how the authentication went.

AV *x y*

Adjust view by setting the horizontal scrollbar to *x* as a fraction of its full distance, where 0.0 is all the way to the left and 1.0 is all the way to the right. The vertical scrollbar is similarly set based on *y* where 0.0 is all the way to the top and 1.0 is all the way to the bottom.

CC ***|***user target message-ID***

Clear the chat history. If the parameter is "*", do so without further comment. Otherwise, the client may inform the user that the chat was cleared, and further if the parameter is not the empty string, it indicates the user name of the person who cleared it.

If *target* is given and not the empty string, it may be a message ID in which case all messages with *message-ID* less than *target* are removed from the history; it may also be a negative integer −*n*, in which case all but the most recent *n* items are deleted.

The *message-ID* parameter is a sequence ID for this command in a manner identical to the **TO** command.

CLR *id*

Remove object with internal ID *id* from the map. The *id* may also be one of the following special values:

*****	Remove all objects from the map.
E*	Remove all map elements (non-creatures).
M*	Remove all monster tokens.
P*	Remove all player tokens.
[*imagename***=**]*name*	Remove the creature with the given *name*.

CLR@ *id*

Retrieve the map file with the given *id* from the server (as the "**M@**" command does) but rather than merging its contents with the map, the client is to delete the

objects described in the file from the local map as if "**CLR**" were called for each of them.

CO *state*
: Set combat mode on or off based on the boolean value *state*. This value SHOULD be 0 for off and 1 for on, but clients MAY interpret any non-zero value to turn the combat mode on.

CS *abs rel*
: Update the client's clock.

  abs The absolute time as a real number of seconds from the GMA clock's epoch.

  rel The real number of seconds which have elapsed from some GM-designated point of reference (usually from the start of going into initiative).

D *recipient-list dice*
: Roll the dice described by the *dice* parameter, sending the results to all of the authenticated users in *recipient-list*. This list may include the usernames of other logged-in users. The list may also include any of the special symbols:

  @ Send back to the user issuing this command.

  * Send to all clients. The presence of this in the *recipient-list* overrides everything else in the list.

  % Send privately to the GM, and *only* to the GM (i.e., do not send a copy back to the user making the die roll, as would normally be done). The presence of this in the *recipient-list* overrides everything else in the list.

DD *definition-list*
: Define a personal set of die roll presets. Each entry in *definition-list* describes a preset and contains three elements:

  name The name of the preset.

  description A long description of what the preset is for.

  dice A list of die roll specifications as appropriate for the **D** command.

  All existing presets for the authenticated user are replaced with the set given to this command. This will trigger a **DD=** message to be sent to all peers other than the requesting connection which are logged in with the same username.

DD+ *definition-list*
: Just as **DD**, but rather than replacing the existing list, the elements of *definition-list* are added to the existing list. This will trigger a **DD=** message to be sent to all peers other than the requesting connection which are logged in with the same username.

DD/ *regex*
: Deletes all elements from the die-roll preset for the requesting user whose names match the regular expression *regex*. This will trigger a **DD=** message to be sent to all peers other than the requesting connection which are logged in with the same username.

DD=
: Begins a dump of defined dice rolls in response to a **DR** command. It is followed by a number of **DD:** commands, and ultimately by a **DD.** command at the end of the list. The **DD=**, **DD:**, and **DD.** commands are not sent by clients, only by the server.

DD: *i name desc dice*
: Gives a preset dice roll definition. The fields are:

  i The position in the list, starting with 0.

  name The name for this preset.

  desc The description for this preset.

  dice The list of dice rolls to make.

DD. *count checksum*
: Ends the list started with **DD=** in the same manner as the **LS.** command (see below).

DENIED *message* Access to the server is denied. The server will send this to a client; no client should send it.

DR Retrieve the authenticated user's die roll presets. This should result in the server sending the **DD=** command back to the requesting client.

DSM *condition shape color* [*description*]
Define a new status marker for creatures with the named *condition*. If there was already a marker defined for that condition, it is replaced with the new definition. If either *shape* or *color* is the empty string (noted as "**{}**" in this data format), then the condition is effectively removed from the list of conditions known to the mapper. Creature conditions are in effect if their name appears in the list value for that creature's **STATUSLIST** attribute as documented in **mapper**(5).

condition This is an aribtrarily-named string that describes the creature's condition that will be noted with a special mark on their token. The mapper comes with the following conditions pre-defined: **ability drained**, **bleed**, **blinded**, **confused**, **cowering**, **dazed**, **dazzled**, **deafened**, **disabled**, **dying**, **energy drained**, **entangled**, **exhausted**, **fascinated**, **fatigued**, **flat-footed**, **frightened**, **grappled**, **helpless**, **incorporeal**, **invisible**, **nauseated**, **panicked**, **paralyzed**, **petrified**, **pinned**, **poisoned**, **prone**, **shaken**, **sickened**, **stable**, **staggered**, **stunned**, and **unconscious**. Remember that values with embedded spaces need to be surrounded by braces (e.g., "**{ability drained}**").

shape The shape of the marker to be placed if the creature has this condition. The mapper will attempt to arrange multiple markers with the same shape such that they are all visible at the same time. This value may be one of the following:

|v A small downward-pointing triangle against the middle of the left edge of the token. (This is a lower-case "v".)

v| A small downward-pointing triangle against the middle of the right edge of the token. (This is a lower-case "v".)

|o A small circle against the middle of the left edge of the token. (This is a lower-case "o".)

o| A small circle against the middle of the right edge of the token. (This is a lower-case "o".)

|<> A small diamond against the middle of the left edge of the token.

<>| A small diamond against the middle of the right edge of the token.

/ A slash (upper right to lower left) through the entire token.

**** A back-slash (upper left to lower right) through the entire token.

// A double slash (upper right to lower left) through the entire token.

**** A double back-slash (upper left to lower right) through the entire token.

– A single horizontal line drawn through the center of the entire token.

= A double horizontal line drawn through the center of the entire token.

| A single vertical line drawn through the center of the entire token.

|| A double vertical line drawn through the center of the entire token.

+ A cross drawn through the entire token.

A hash-mark drawn through the entire token.

V A large downward triangle drawn around the entire token. (This is an upper-case letter "V".)

^ A large upward triangle drawn around the entire token.

<> A large diamond drawn around the entire token.

O A large circle drawn around the entire token. (This is an upper-case letter "O".)

color The color to draw the marker in any of the forms documented above, or the special value "*****", which means to draw the marker in the same color as the creature's threatened area.

If *color* begins with "**−−**" then the marker is drawn with dashed lines instead of solid ones. If it begins with "**..**" then the effect is the same, but the dashes are shorter.

description If present, this field describes the effect of the condition on a creature.

GRANTED *name* Access to the server is granted. The server will send this to a client; no client should send it.

I *time id* Update the time clock for initiative-based actions. If the mapper is in combat mode, the clock display is updated accordingly. The *time* parameter is a list of five values, in this order:

r Number of rounds elapsed since start of combat.

c The initiative count within the round. This is a number from 0−59, which indicates which of 60 possible initiative slots is currently active. Since this means we are effectively dividing the round into 60 equal parts, each "count" represents 0.1 seconds of game time.

s Number of seconds elapsed.

m Number of minutes elapsed.

h Number of hours elapsed.

The *id* parameter gives the name of the creature whose turn it is. This may be the name of a player or monster (exactly matching one of the values given by the **−c** option(s), or creatures added via "**AC**" commands), the unique ID for a creature, the special string "***Monsters***" (to indicate that all tokens of type "monster" get to act now), or the empty string (written as "**{}**" due to the rules for Tcl list string representation) which means no one has initiative at this point in time.

Alternatively, if *id* begins with a slash ("**/**"), then all creatures whose names match the regular expression *id* (not including the leading slash) are selected.

If the mapper client is not in combat mode, this command has no effect.

IL *slot-list* Update the initiative list. The *slot-list* parameter given is a list of an arbitrary number of sub-lists. Each sub-list consists of the following six values which describe a

combatant:

name The creature's name as it appears on the map. This must be unique with respect to the other creatures listed. It may have the forms acceptable to the "**AC**" command.

hold Boolean value; if true, this person is holding their action.

ready Boolean value; if true, this person is not only holding their action (and thus the *hold* field should also be true) but have a readied action waiting to trigger.

hp Their current hit point total.

flat Boolean value; if true, this person is flat-footed.

slot The slot number in the initiative list for this combatant. (An integer from 0–59 as described above.)

L *filename-list* Load one or more map files into the mapper. The current contents of the client's map canvas are to be deleted, then the named *file* loaded from disk. Its contents are as documented in **mapper**(5). The objects described in the file are added to the local map. For completeness (and consistency with the "**M**" command), multiple files may be named, but since the map contents are cleared before each one, only a single file should be specified with this command. The files must be accessible from the local mapper process (and relative pathnames are relative to its current working directory). The braces are merely illustrative in this case. The argument to this command is a TCL list of filenames.

LS Begin loading an object from the data stream rather than from a map file. This is analogous to the "**AI**" ... "**AI:**" ... "**AI.**" sequence of commands described above.

LS: *data* The map data are loaded from the sequence of "**LS:**" commands as if read from a local map file, one line of data (i.e., one object attribute record) per command.

LS. *count checksum* Following the last data packet, this line terminates the stream transfer. See "**AI.**" for a description of how this command ends the data sequence. The *checksum* value is calculated as the SHA-256 hash of the *data* values of each command.

M *file-name-list* Merge one or more map files into the mapper. This is just like the load ("**L**") command, except the contents of the file are *added to* the existing mapper contents rather than replacing them. Since the argument to this command is a Tcl list, braces enclose it if there is more than one filename. Examples:

M room12.map
M {room1.map room2.map room3.map}
M {room4.map {south corridor.map}}

M? *id* Ensure that we have a locally-cached copy of a map file with the given *id*. This performs an operation similar to that of "**AI@**" (looking for a file with a "**.map**" suffix on the server) in that it checks to see that a cached copy exists. If it does but isn't recent enough, it will check the server to see if a newer version exists there. If no cached version is found or the server has a newer one, the file is downloaded to the cache. It does not actually load the map file, however. See "**M@**". The *id* has the same form as that allowed for "**AI@**".

M@ *id* Identical to "**M?**" but will then merge the contents of the map file with the current map data in the same fashion as the "**M**" command does.

MARK *x y* Make a brief animated marker at the specified coordinates to draw attention to that space.

MARCO Status check. If received, reply with a **POLO** command. **N.B.** It is important that your client not ignore these occasional ping messages. For example, if your client

is too slow receiving messages such that the server needs to expend extra work to queue them up for you, it will be willing to do so if you have been at least responding to **MARCO** messages. If you haven't, the server will suspect your client has locked up or is not going to be able to catch up with the data being sent to it, and may decide to terminate the client's connection.

NO Request that the server send no more data to the client.

NO+ Same as **NO**, but the client is also asserting primary control role to the other clients. (This is generally done by the GM console. The effect is that the server sends a "**TB 0**" command to tell all mapper clients to turn off their toolbars (unless, of course, they were invoked with the **−−keep−tools** option) and the server will automatically tell all clients which connect after this point to start with their toolbars disabled as well.) Once all "primary" clients (those who issued this command) have disconnected, the server will stop telling new clients to turn off their toolbars. *This is deprecated. In the future, the concept of a primary client will not be supported by servers, and management of toolbars will be at the clients' discretion.*

OA *id kvlist* Update a set of arbitrary attributes for the object with the given *id*. The *id* is either the unique identifier for the given object, or has the form @*name* where *name* is the name of a creature in any of the forms accepted by the "**AC**" command.

The *kvlist* parameter is a list with an even number of elements. The first gives the name of an attribute. The next gives a new value for that attribute, and so on by pairs to specify a set of attributes to change.

For example, the command

OA 12345 {X 13 Y 47}

alters the **X** and **Y** attributes of the object with ID **12345** to the values 13 and 47, respectively. For a typical map element, this will move the object so its reference point is at coordinates (13, 47).

Note that there was an implicit assumption in the past that the **NAME** attribute would not change and it could be used as an immutable identifier within a map client for a creature token. However, this is not the case and in fact GMA now includes explicit features to change this very attribute. Clients must be prepared to deal with the consequences of a change to any attribute, including **NAME**. The **mapper** client itself does this correctly starting with version 3.39.

OA+ *id key vlist* Assuming that the object with the given *id* (which may be specified as described for the **OA** command) has an attribute *key* whose value is actually a list of values (e.g., the **STATUSLIST** attribute of creature objects), this command adds the values from *vlist* to that attribute's list. If any value was already in the attribute's list, it is not added again.

OA− *id key vlist* Just like **OA+** but removes the specified values from the attribute's list rather than adding them. It is not an error if any of them did not already exist.

OK *v* [*challenge*] The server sends this to the client after the initial set of messages that are sent upon connection. It signals that those messages are over and the server is now ready to listen to the client. The *v* value is an integer which indicates the protocol version supported by the service. Map clients which do not support that protocol should warn their users to upgrade.

If present, the *challenge* value is a base-64-encoded authentication challenge. A client must successfully respond with a valid **AUTH** command before any of its commands to the server will be accepted. The server will also refuse to send the client any map updates until it has successfully authenticated.

PRIV *message* Notice from the server that a command sent by the client is denied due to insufficient privilege. Clients should not send this command.

POLO No-op. Ignore if received.

PS *id color name area size* **player|monster** *x y reach*
Place a creature token on the map. This may be used to define a new creature object if no object with the given *id* already exists, or replaces an existing token with the (possibly different) parameters given.

id The internal ID by which this creature is to be known. This must be unique. The client which creates the character token locally should create a unique ID, which is then broadcast via this command to the other clients, which use the same ID.

color The color used to shade the creature's threat zone when they have initiative to act. It has the same forms as accepted by the "**AC**" command.

name The name of the creature as will be displayed on the token. It has the same forms as accepted by the "**AC**" command.

area The size of the threat zone around the creature. It has the same forms as accepted by the "**AC**" command.

size The size of the creature itself. It has the same forms as accepted by the "**AC**" command.

type This is either the constant string "**player**" or "**monster**", to indicate whether the creature is generally considered an ally or adversary (respectively) to the players. The only effect this has is that creatures of "**monster**" type may be selected by the special token "***Monsters***" when specifying whose turn it is in initiative, and are deleted by the "**CLR M***" command, while player type creatures are deleted by the "**CLR P***" command.

x The location on the *x* axis for the creature's reference point (upper-left corner). **Note** that this is in *grid units*. The leftmost square of the map is 0, the next square to the right is 1, and so forth. Creatures smaller than "small" size category may be placed with fractional values such as 0.5 for the right half of the left-most square on the map.

y Like *x*, but along the *y* axis.

reach Boolean value; if true, the creature is wielding a reach weapon and thus has an extended threat zone.

ROLL *from recipient-list title result structured-result-list message-ID*
Reports the results of a die roll initiated by the **D** command. The values given are:

from The user who initiated the roll.

recipient-list The list of users to receive this notice, in a form identical to the **TO** command.

title A descriptive title for the die roll. This may be the empty string.

result The numeric result of the die roll.

structured-result-list A list of tuples describing the actual die rolls that resulted in the reported *result* value. See **dice**(3) for a description of what this list may look like.

message-ID A server sequence for the message as described for the **TO** command.

SYNC [**CHAT** [*target*]]

This command is *not* recognized by any client (if a client receives one, it should simply ignore it) and is not relayed by the server to any other clients. Instead, this is a request by a client to the server itself. The client should completely clear its internal state before making this request. In response, the server will send to the client its understanding of the current state of the game. This should catch up the client to match what the other map clients are seeing.

If a parameter with the word **CHAT** is added, then the chat message history is synced to the client instead of the map contents. In this case the optional parameter *target* may be used to get only a subset of the chat history. If *target* is a negative integer $-n$ then only the most recent n messages are sent; otherwise it is assumed to be the *message-ID* the client last saw, so only messages with IDs higher than *target* are sent.

A server may be configured to perform a sync operation upon client connection (after, if required, authentication) without the client explicitly sending a "**SYNC**" command to it.

TB *state*

If *state* is a boolean true value, display the toolbar in the mapper client, otherwise hide it. A client may choose to disregard this command.

TO *from recipient-list message message-ID*

Send chat message to a set of recipients. The values in the recipient list are as described above for the **D** command. The *from* parameter is ignored when received from a client and may be the empty string. It is replaced by the username of the sender when the server resends the command out to all clients. The *message-ID* value is ignored when sent by clients, but when the server relays this out to the peer clients, it will add a unique (and sequentially increasing) value to identify the message. Clients can use this to request updates via the **SYNC CHAT** command or to sort messages chronologically.

/CONN

For diagnostic purposes, this command causes the server to send a summary of the connections it is currently serving. These are sent back to the client using the **CONN** ... **CONN:** ... **CONN.** command series. It is not passed on to peer clients.

CONN

Begin a sequence of data records describing current connected clients. This is followed by a number of **CONN:** commands.

CONN: *i* **you|peer** *addr user client auth? primary? w/o? polo*

Gives one data record describing a connection to the server. The fields included are:

i	Position in the data list, starting with 0.
you|peer	Indicates if this describes the requesting client or a peer connection.
addr	The IP address of the connection.
user	The username given by the client when logging in.
client	The client tool using that connection.
auth?	A boolean value, true if the connection authenticated successfully.
primary?	A boolean value, true if the connection asserted primary mode.
w/o?	A boolean value, true if the connection is in write-only mode (will send commands to the server but it not listening for any replies).
polo	The number of seconds (which may be an integer or real number) since the last **POLO** command was received from this client. This may help debug stuck clients or dropped connections.

CONN. *count checksum*

Marks the end of the sequence of data records, just as the **LS.** command does (q.v.). The *checksum* value may be 0, in which case there is no calculated checksum provided.

WEIRD SIZES

While the mapper implements the standard d20/Pathfinder creature size categories, including tall (uppercase) and wide (lowercase) variants, sometimes there are special cases which fall outside that list. The following special codes are also usable:

M20/m20 Medium creature (5-foot space, 5-foot threat zone) with a 20-foot reach zone.

L0/l0 10-foot swarm (10-foot space, no threat or reach zone).

INSTALLATION

To run the mapper, you'll need an up-to-date Tcl/Tk interpreter (8.6 or later), and the following packages in your Tcl runtime library:

- uuid
- base64
- struct::queue
- tklib

Additionally you will need a copy of **curl**(1) installed on your system.

If you will be uploading content to the web server (and are authorized to do so), you will also need to have **ssh**(1) and **scp**(1) on your system.

You will need to ensure that the paths to these commands, server name(s), data paths, etc, are configured correctly for your needs as well.

SEE ALSO

curl(1), **scp**(1), **ssh**(1), **dice**(3), **dice**(5), **mapper**(5), **gma**(6).

The wiki page **https://www.madscience.zone/gamers/wiki/GmaSoftwareDevelopment/Protocols/MapperProtocol** contains more detail on the map protocol, but this manpage is the definitive reference to the protocol.

AUTHORS

Steve Willoughby / steve@madscience.zone; John Mechalas (elevation and movement modes).

HISTORY

This document describes version 3.39 of **mapper**. (A previous version of this manpage erroneously listed the mapper version as 4.) A version of **mapper** was also in version 3 of GMA, but was different in operation.

As of version 3.25, the operation of the "Push to other clients" button was changed so that it only works in store-and-forward mode (and is thus reserved essentially for privileged users only). This was done because the old function of that button is no longer needed and tended to cause more trouble than it was worth anyway.

Also changed in 3.25 is the function of the "Sync" button. It used to simply attempt to reconnect a client to the server (which should automatically happen anyway). Now, since the server tracks game state, simply exiting and restarting the map client accomplishes the same effect (possibly better). Now this button requests a "sync" operation with the server.

This document also describes map protocol version 333, which added the **ALLOW** command.

Version 332 added the **GRANTED** server response, guaranteeing that **AUTH** will be followed by a response in any case.

Version 331 changed the way persistent chat messages are managed, altering the **ROLL**, **TO**, **SYNC**, and **CC** commands.

Version 329 added the **CC** command and the extended form of **SYNC**.

Previously, version 328 introduced the **ACCEPT** command.

Version 327 added the **DENIED** and **PRIV** response commands.

Version 326 added support for chat channels and die rolling, and changed the response from the **/CONN** command to be a structured sequence of data records rather than sending the reply in comments.

Protocol 325 added connection debugging support by changing the **AUTH** command and adding **/CONN**.

The version 324 protocol added the **DSM**, **OA+**, and **OA−** protocol commands. This version of the protocol also assumes mapper file format 15 is being used.

Previously, protocol version 323 added support for server-side persistent state and the introduction of the **SYNC** protocol command.

Protocol version 322 added comments to version 321.

Protocol version 321 added authentication support to the capabilities of protocol 320.

Map protocol version 320 has the same commands as 319, but the **HEALTH** attribute of creatures changed from map version 12 to 13, so we are advancing the protocol version at the same time since maps released for protocol 319 were expecting the older map format.

Map protocol version 319 differs from version 318 (the first explicitly numbered version) by the addition of the **CLR@**, **M?**, and **M@** commands.

COMPATIBILITY

This program requires a reasonably modern version of Tcl/Tk, tcllib and tklib to function properly. We strongly recommend running it with the latest versions of all of those.

It is known to run on the macOS Mojave platform (tested on 10.14.6), macOS Catalina (tested on 10.15.3, but note that Apple's support for python3, tcl, and tk is such that you may want to install your own versions of those tools); and should run fine on any modern *NIX-like platform (tested on FreeBSD 12.0, Ubuntu 18.04 LTS, and Ubuntu 16.04 LTS).

It was also tested (briefly) on Windows 10. Specific notes about using it on that platform are in the following section.

WINDOWS USAGE

To install on Windows, you will need to install the following products in addition to GMA:

- A Tcl/Tk interpreter. We tested with Active Tcl 8.6 from *activestate.com*.
- The Curl utility. We tested with Curl 7.67.0 from *curl.haxx.se*.
- The SSH shell program, including the SCP command for copying files (if you want to use the store and forward mode to send data to the server; note that this requires authorization to perform that action by the server administrator and that you have configured the SSH/SCP program(s) with the valid credential certificates). We tested with PuTTY 0.73 from *www.chiark.greenend.org.uk*.

By default, the **mapper** program will try to load its configuration file from *homedir***\.gma\mapper\mapper.conf** if that file exists, where *homedir* is the user's home directory. This allows the mapper to be launched by double-clicking its file icon without needing to provide command-line options.

We recommend that you set up this configuration file before starting, since we don't currently have a fancy Windows installer wizard thingy to configure it for you.

In this file, you will need at a minimum to tell the mapper where to find the **curl** program and the network locations in use for your game. For example:

```
curl-path=C:\curl-7.67.0-win64-mingw\bin\curl.exe
curl-url-base=https://www.example.org/game/map
host=www.example.org
port=2323
proxy-url=http://proxy.example.org:1080
```

If your game does not use a network server at all, it's possible you won't need a configuration file.

Handling of standard output and standard error devices for printing messages is not as well-supported in Windows as in Unix-like operating systems, so the mapper will attempt to place output in files called **mapper.***pid***.stdout** and **mapper.***pid***.stderr** in the *homedir***\.gma\mapper\logs** directory. If the mapper just silently fails to start, look in those to see if there were fatal error messages recorded there.

The mapper does not currently support being the forwarder in a store and forward configuration for Windows clients.

FILES

~/.gma/mapper/mapper.conf
: Default configuration file read if no explicit **−C** or **−−config** option is given.

~/.gma/mapper/style.conf
: Default custom style configuration file read if no explicit **−s** or **−−style** option is given.

~/.gma/mapper/debug.log
: Location where debugging messages are written in addition to being displayed in the debugging window.

~/.gma/mapper/logs
: Runtime logfiles are stored here for each execution of the mapper.

~/.gma/mapper/cache
: Cached copies of images and other content are stored here to improve speed of the mapper.

BUGS

There are numerous hacks in the program which really should not be there. In fact, at this point the thing just needs to be rewritten using the newer GMA code base.

Calculation of threatened spaces needs to take elevation into account.

Arrowheads need to be drawn bigger.

The **−h** option really should have been for **−−help** to conform to usual command-line conventions, and the **−−host** option should instead have been **−H**. This may change in the future.

In previous versions, **−−keep−tools** (**−k**) was called **−−master** (**−m**), but this never really made sense, as it didn't really mean the mapper was in any sort of controlling or leadership role. It only meant it would refuse to turn off its own toolbar if asked to do so. The new name is more descriptive of the actual function.

COPYRGHT