# MUD Sound Protocol (MSP)

The MUD Sound Protocol (MSP) was developed jointly between many people, including implementers of various MUDs and MUD clients. Leading the MSP design effort was Medievia MUD and the zMUD and MUD Master MUD clients. This document provides the complete MSP specification, as well as zMUD implementation notes. This document was cleaned up by Tomas Mecir.



## MSP Specification v0.3 (updated)

**Changes from v0.2:**

Added U=URL optional parameter for !!SOUND and !!MUSIC to download sounds via the web

**Protocol basics:**

The purpose of the MUD Sound Protocol (MSP) is to allow different sounds/music to be played in certain situations. To achieve this, both server and client must support this protocol. The protocol itself is really simple. The server can send special sequences, called **sound triggers**, that result in sound being played at client side. No information, requests or anything is sent in the opposite direction. The protocol also includes support for downloading sound files from the server.

**Protocol negotiation:**

Server can only send sound triggers if it knows that they will be recognized and processed by the client. This is achieved by standard telnet option negotiation (see RFC 854 for more information).

MSP is assigned option number **90**. (note that this is not "officially" assigned by standard authority, so it's possible, though very unlikely, that this could collide with some other option).

**IAC WILL MSP**
MSP negotiation sent by the **server** to detemine if the client supports MSP. After receiving IAC DO MSP from the client, the server is free to send sound triggers to the client.

**IAC WONT MSP**
Sent by the **server**, informing the client that it will no longer send sound triggers. Should only be sent as a response to client's IAC DONT MSP. See RFC 854 for more information on when to send IAC-strings to avoid endless loops.

**IAC DO MSP**
Sent by the **client**, as a response to server's IAC WILL MSP negotiation string. In some situations, this can also be sent spontaneously, see below.

**IAC DONT MSP**
Sent by the **client**, informing the server that it should no longer send MSP triggers. Also sent as a response to negotiation sequence, if the client does not support MSP, or does not want to support MSP.

If the server does not implement telnet option negotiation, then it should implement a command that the player is required to enter in order to enable MSP.

**Disabling and re-enabling MSP:**

Client may want to allow users to disable MSP, even if it's already negotiated. There are two possible solutions, it's up to the client to choose one of them.

**Solution 1**
Disable MSP locally in the client, but do not notify the server. Server will continue sending sound triggers, but they will be ignored by the client.

**Solution 2**
Disable MSP using telnet negotiation (IAC DONT MSP), then re-enable it if desired (IAC DO MSP). This approach is recommended if the server properly implements the telnet option negotiation described above.

**Sound triggers:**

Sound triggers can only be sent by the server. Each sound trigger must be located on the start of a line and followed by a CR/LF (i.e. each sound triggers is on its own line). If a sound trigger is found in the middle of a line, it should be treated as ordinary text. This is to prevent users to sent malicious sound triggers.

A client may allow sound triggers in the middle of the line, but this should be made optional with default off, to eliminate potential risks for the end-user.

**Format of sound triggers:**

There are two types of triggers - **sound triggers** and **music triggers**. Each trigger consists of two parts - **type string** and **parameters**.

**Type string**
identifies the trigger and its type. Type string is

```
!!SOUND
```

for sound triggers. Sound triggers are WAV format files intended for sound effects. Some clients can play several sound effects at once, but other clients can only play a single sound effect at a time and might stop playing the previous sound if another sound trigger is sent before the client is done playing.

**!!MUSIC**

for music triggers. Music triggers are typically background MID (MIDI) files. Only one music trigger can be active at once. Some clients (such as zMUD) send the file directly to the operating system to be played, so any file format that is supported by the operating system is valid. But MID files should always be supported by the client.

**Parameters**
immediately follow the type string, with no delimiter between them. Parameters are enclosed in parentheses '(' and ')'. The individual parameters are then separated by one or more spaces. Therefore, a space cannot be a part of a parameter. There is always at least one parameter.

**First parameter**
is called **fName**, it specifies a file name. It may contain relative path information and wildcards, but may not contain absolute path information. Any directory information should be specified using "/" rather than "\", since this avoids all of those nasty problems with escape characters. If multiple wave files match a wildcard, one of them should be chosen at random. Wildcards should work just like DOS wildcards; * should match the remainder of the file name, and ? should match exactly one character. If no extension is specified, ".wav" should be assumed for SOUND triggers, ".mid" for MUSIC triggers.

If a wildcard pattern with multiple matching files is received, and number of repeats is greater than 1, the actual file name should be picked randomly for each iteration.

File name should be transmitted as case-sensitive, but the server must also ensure that no conflicts shall occur on a case-insensitive OSes. Therefore, if there is a file named FilE1.wav, server must send its name as File.wav, not file1.wav or snything similar, and there mustn't be any file named file1.wav, File1.wav and similar names in the same directory.

Directory information can be used to maintain a hierarchy of sounds. When looking for a sound file, a client should look in the given subdirectory of its sound directory. It can also ignore the path information and look directly in sound directory, but this should only occur if a sound has not been found in the first place. Only relative file paths are allowed. Attempts to specify an absolute path should be ignored by the client.

File name **Off** is reserved. When a sound/music trigger with this fName and no further parameters is received, playing sound or music file(s) (depending on trigger type) should be stopped immediately.

**Following are some valid fName values:**
weather/thund*
lightning.wav
zone231/room22.wav
**And some invalid ones:**
weather\lightning.wav (uses \ rather than /)
*x*.wav (wildcards stop after the first *)
c:\sounds\m.mid (no absolute paths)

**Following parameters**
are optional. Each of them has a default value that is used if the parameter is not given. Format of these parameters: <param-name>=<param-value>, where param-name is a one-letter upper-case identifier (A-Z).

**Recognized parameter names**
are as follows. Most of them can be used with both SOUND and MUSIC triggers, some are only valid for one of them.

**V parameter**
Specifies sound/music volume for this request.
Possible values: **0-100**
Default value: **100**
Used in: **SOUND, MUSIC**

**L parameter**
Specifies number of repeats. The sound/music file should be played this many times. It can also be -1, which means that the sound/music file should be played infinitely, until instructed otherwise.
Possible values: **-1, integers > 0**
Default value: **1**
Used in: **SOUND, MUSIC**

**P parameter**
Specifies sound priority. This parameter applies when some sound is playing and another request arrives. Then, if new request has higher (but NOT equal) priority than the one that's currently being played, old sound must be stopped and the new sound starts playing instead. In the case of a tie, the sound that is already playing wins. This is to prevent combat sounds from degenerating into a mishmash of sound fragments.
Possible values: **0-100**
Default value: **50**
Used in: **SOUND**

**C parameter**
means "continue", specifies whether the file should simply continue playing if requested again (1), or if it should restart (0). In either case, the new repeat count should take precedence over the old one, and the "number of plays thus far" counter should be reset to 0. By way of illustration, assume two rooms, room 110, and room 111. Room 110 is set to play bach/fugue.mid 3 times, while room 111 is set to play bach/fugue.mid 5 times. If a character enters room 110, fugue starts playing; if during halfway through the second refrain the character moves to room 111, fugue would either continue or restart based on the continue setting. fugue should play either 4.5 times (if continue was 1) or 5 times (if continue was 0). Similarly, the volume of the most recent MUSIC escape should be used.
Possible values: **0, 1**
Default value: **1**
Used in: **MUSIC**

**T parameter**

Specifies type of the sound. This is mud specific. Some example classes might be combat, zone, death, clan. This parameter may be omitted, but if a MUD opts to use it, it's recommended that it be used everywhere. The actual type is a string, and should be case insensitive. It's up to the client to determine whether to use this parameter, and how to use it. This parameter was intended to provide a way to group sounds into subfolders within the main sound directory. Although subdirectories can be specified in the filename itself, the T parameter is another way to accomplish this.
Possible values: **any string**
Default value: **subdirectory of fName**
Used in: **SOUND, MUSIC**

**U parameter**

This is an optional parameter and is not supported by all clients. This parameter specifies the URL of the sound/music file. This allows downloading files from the MUD server. Client should always look in local directories first, and only download the file if it's not available locally. Each client supporting this option should support at least the **HTTP** protocol. Other protocols can be supported as well. The URL of the sound file is determined by combining the value of this parameter and fName. E.g., if fName is *combat/win1.wav* and URL is *http://www.example.net:5000/sounds/*, the URL of the remote file is *http://www.example.net:5000/sounds/combat/win1.wav*. The '/' character should be appended to U param if not given. The URL can also be enclosed in quotes (").

It is possible to specify default URL for all requests, that will be used in all subsequent request without the need to specify it in every trigger. This can be accomplished with a special type of trigger, with fName=Off and U param given. This URL will then be used as a default URL. See below for an example.

Clients should provide an option to allow users to turn off sound downloading, since this can have a significant impact on MUD playing performance. The original intent of MSP was to require the MUD to bundle its sound files and to require the player to download and unpack the sounds used by the MUD before playing. The U parameter was added later to allow on-the-fly downloading of sounds for players with fast network connections.

Possible values: **URL string**
Default value: **No URL, uses previously set default URL. If no default is set, sound is not downloaded**
Used in: **SOUND, MUSIC**

**Examples:**

```
!!SOUND(thunder V=100 L=1 P=30 T=weather)
!!SOUND(weather/rain.wav V=80 P=20 T=weather)
!!SOUND(alarm*.wav P=100 T=utility)
!!SOUND(Off)
!!SOUND(Off U=http://www.example.org:5000/sounds)
!!MUSIC(fugue.mid V=100 L=1 C=1 T=music U=http://www.example.net/)
!!MUSIC(berlioz/fantas? V=80 L=-1 C=1 T=music)
!!MUSIC(Off)
```