Portal© for Windows MIP v9.0

(Messaging Interface Protocol)

Use of this document in any way deems acceptance of the Terms of Use License. This policy can be located at the bottom of this document. If you do not agree to the conditions outlined in the Terms of Use License, you may not use this document, nor the intellectual property contained therein.

The term MIP (Messaging Interface Protocol) is used to describe the means of communicating with Portal© through the sending of text strings. That said, let's get right into the coding specifics.

The **3klient** Command

The security code (*SEC_CODE*) is a 5-digit number that is generated randomly by Portal©. This code is somewhat like a PIN number. It prevents bogus messages (those that don't match the code currently held locally by the Portal© user) from being recognized by Portal©. The security code must be sent with any data coming to Portal©. It can be anywhere between 1 and 5 digits in length (between 1 and 99999). The security code will be sent to your mud from Portal© through a command, namely '3klient #~<version>' (~ separates the code from the version info) such as '3klient 10401~Portal3.500', so you need to accept the number and store it as a static variable on the player. The text following the tilde is simply version info. You can either parse it as you desire (for a client-who list for example) or simply ignore it. You create the 3klient command with an add_action() function or however you normally create commands for characters on your MUD. This provides the basis for the Portal© "jumpstart" feature, which sends the 3klient command. The "3klient" command is in all lowercase letters.

Remember, the *SEC_CODE* number you send to Portal@ in any code must be 5 characters in length. That means that if the number a player has attached with is 675, the *SEC_CODE* you would always send that player is 00675. With the jumpstart the player sends though, the numbers are not prefixed. This means that a jumpstart sent by the player in the example just stated would look like '3klient 675~Portal3.500' The reason for the difference is that it makes it easier for MUDs to grab the *SEC_CODE* (via a sscanf function for example).

Notes: 3Klient? What the hell is that? Well, at one point in time, this MUD client was called 3Klient. That name was purged and replaced by the current name – Portal©. Since the trigger command needs to be something universally unique, we decided to leave 3klient as the trigger, both because of its uniqueness, but also as a reminder of its heritage.

The **Welcome** Trigger

Now, the player can do the jumpstart manually, or your MUD can trigger it for them. We have created a resident trigger in Portal© that respond to the word "Welcome" appearing on the screen. When Portal© connects to your MUD, near the end of their normal login spam, put the phrase "Welcome" into a greeting line. Something like "Welcome to SuperMUD!" would work. Once Portal© sees the "Welcome" in a line, it will send the jumpstart "3klient" command automatically. It will also de-activate the trigger until the next, fresh login (to prevent the jumpstart being sent each time the word "Welcome" comes across the screen). Once this is done, the 3klient command, which you setup as described above, will attach your user to your mud accordingly.

Notes: v1.4 does not have the Welcome trigger. They must manually jumpstart.
v2.0 has the trigger, but requires something on either side of the "Welcome" Any character will do such as: "Hello! Welcome to BlahMUD!"
v3.0 actually triggers on text defined by the user in the MUD List screen The default event text is "elcome" but again, is configurable by the user. The default will change if that MUD desires it (and contacts us).

How does the extra information get sent without the player seeing it?
To send information to Portal© there's only a few special strings and characters that you need to know, we'll refer to these as literals. The most important literal is what as known as the *ACTIVATE* literal. This must precede all information sent to Portal©. Its value is **#K%**. Following the *ACTIVATE* literal will be the **Security Code** (*SEC_CODE*), the **Character Count** (*CHAR_COUNT*), the line code **Literal** and the **Data** you wish to send. This will be discussed in more detail below.

ACTIVATE+*SEC_CODE*+*CHAR_COUNT*+**Literal**+**Data**
or
#K%+**Security Code**+**Character Count**+**Literal**+**Data**

To recap, when sending information to Portal©, send the *ACTIVATE* literal first (#K%). Note, do not precede or follow the literal with a line feed.

The next piece of information you will send when communicating with Portal© is the *SEC_CODE*. This is the 5-digit code that hooks up the server (MUD) with the client (the Portal© running on the user's machine). The 3klient command is used to synchronize these values as described above. Also, as described above, it must be 5 digits, so a smaller number would be prefixed with 0s, such as 01459 or 00002.

The third piece of information you will send is the *CHAR_COUNT*. This value will be the total number of characters following it that Portal© needs to intercept and use. It must be 3 digits, so a smaller number would be prefixed with 0s, such as 031 or 009.

The next thing on the line is the line code literal. This tells Portal© what data is to follow. A list of the valid line code literals is in the Appendix A. All line code literals are 3 characters, such as FFF or AAB.

After the line code literal is the data you wish to send. This, of course, is based on which line code literal you choose. If you use the hit point line literal you will be sending hit point and other related information after it. If it is a mud chat, or an image, appropriate information should be sent as well. Individual fields within the data stream must be separated with the *CL_DELIM* literal. Its value is ~ (a single tilde), and should be used to separate all types of information for line code literals that accept multiple values (such as FFF).

Here are some examples of what gets set to Portal©:

#K%00005008AAC12:46
#K%00005008AAF00:50

These two examples break down as follows:
#K% The *ACTIVATE* literal
00005 My security code
008 The number of characters to follow
AAC This is the line code literal for displaying when the mud will next reboot
AAF This is the line code literal for displaying how long the mud has been up
12:46 This is the data. In this example the mud will reboot in 12 hours and 46 minutes.
00:50 Data as well, indicating that the mud has been up for 50 minutes.

Most line code literals will send a single data value similar to the two examples above. However, there are some that expect two or more values on the line. For example, the chat line literal expects 4 values. These 4 values are the line command, line name, sender's name, and line text, delimited by the *CL_DELIM* (~). So, if a player's guild were Mages, an example of the data for a chat would be:

magechat~Mage Line~Rastafan~[Mages] Rastafan: Hi guys!

So the line would be sent as:

#K%00005057CAAmagechat~Mage Line~Rastafan~[Mages] Rastafan: Hi guys!

Where **#K%** is the *ACTIVATE* literal, **00005** is my security code, **057** is the character count to follow, **CAA** is the chat line code and the data is described above.

Portal©, of course, will display that data in the Chat Monitor when it receives it.

The largest of the multiple-data literals is what is called the *COMPOSITE* literal. This is the one that handles the hit point, spell point, guild points, and related information. This is the only line code literal that can accept a variable number of different data points. Basically just fill it up with however many you want. The data points are identified by the *HP_DATA_CODE* literal that is part of the data stream. All *HP_DATA_CODE* literals are a single character. A list of the *HP_DATA_CODE* literals is in Appendix A.

An example of building the line code literal of a **COMPOSITE** is built as follows:

- The *ACTIVATE* literal
- The *SEC_CODE* variable
- The *CHAR_COUNT* of the data
- The *COMPOSITE* literal (**FFF**)
- The *HP_DATA_CODE* literal for the first data point
- The first data point
- The *CL_DELIM* literal
- The *HP_DATA_CODE* literal for the second data point
- The second data point
- And so on…(if you wanted to add a 3rd composite or further)

The final, formatted *COMPOSITE* comes out looking like this:

#K%00005014FFFA322~B472

#K%	The *ACTIVATE* literal
00005	Again, my security code
014	The number of characters to follow
FFF	The line code literal for *COMPOSITE*
A	The *CL_SEND_HP* literal indicating current hit points
B	The *CL_SEND_MAXHP* literal indicating maximum hit points

In this case there are only two data points, and thus two *CL_SEND_HP* literals **A** and **B**. You can send any number of different *CL_SEND_MAXHP* literals in the *COMPOSITE*.

The first data point is **322** and the second is **472**. **A** is the *CL_SEND_HP* for current hit points and **B** is the *CL_SEND_MAXHP* literal for max hit points. So this example tells the client that the player is currently at **322** hit points out of **472**. This is explained better in Appendix A.

Appendix A: Line Code Literals

Note: For all the below examples, we use 00005 as the *SEC_CODE*

#define *CL_SEND_SOUND* **AAA**

This is a single data point literal.
This literal is used to send the ***filename*** of a sound that you wish Portal© to play. All sounds are .WAV files and are stored in the media/sounds/ directory beneath the dir where the Portal© executable resides.
Range for this literal is restricted by Windows' file naming conventions. As a general rule, you should not use anything but alphanumeric characters (a, b, c, 1, 2, 3) and the underscore (_). The filename is not case-sensitive. The length of the total filename cannot be longer than 255 characters.
Note: This function is only enabled if the user has not disabled system sounds.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_SOUND+**filename***
Example: ***filename*** = lock
Formatted Example: #K%00005007AAAlock

#define *CL_SEND_IMAGE* **AAB**

This is a dual data point literal.
This literal is virtually identical to the *CL_SEND_SOUND* literal except that it is used to send the ***filename*** of the image to be displayed in the Portal© Imagery. All images must be either bitmaps or gifs (.BMP and .GIF extensions respectively). You do not need to include the file extension unless you wish to specify. Portal© will look for a bitmap and then a gif, in that order. All images are stored in the media/images/ directory. The ***imagelabel*** (which will appear below the image in the Imagery) is optional, but the *CL_DELIM* is not. If you don't want a label, simply end the string with *CL_DELIM*.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_IMAGE+**filename**+CL_DELIM+**imagelabel***
Example: ***filename*** = grey_elf
Example: ***imagelabel*** = Gray Elf
Formatted Example: #K%00005020AABgrey_elf~Gray Elf

#define *CL_SEND_REBOOT* **AAC**

This is a single data point literal.
This literal is used to send the amount of time (***timestring***), in hours and minutes, that are left until the mud has a scheduled reboot. The format should be HH:MM.
Note: If the Caption (**CAP**) literal is used, the Reboot literal is not used.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_REBOOT+**timestring***
Example: ***timestring*** = 12:45
Formatted Example: #K%00005008AAC12:45

#define *CL_SEND_MUSIC* **AAD**

This is a multiple data point literal.
This literal is similar to the CL_SEND_SOUND except that it is used to send the ***filename*** of the music file to be played. Valid MIDI files formats are supported (*.MID, .RMI, etc.) as well as MP3 files (.MP3) You do not need to include the file extension unless you wish to specify. All music files are stored in the media/sounds/ directory. You can also choose the number of ***iterations*** to play of the file. The file will play ***iterations*** # of times. If you don't specify a number of ***iterations***, the file will play only once.
Note: This function is only enabled if the user has not disabled remote media support.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_MIDI+**filename**+CL_DELIM+**iterations***
(non-repeating music example)
Example: ***filename*** = canon_in_D
Example: ***iterations*** = <nothing>

Formatted Example: #K%00005013AADcanon_in_D

(repeating music example – plays it three times)
Example: ***filename*** = canon_in_D~1
Example: ***iterations*** = 3
Formatted Example: #K%00005015AADcanon_in_D~3

(stops the currently playing music file)
Example: ***filename*** = <nothing>
Example: ***iterations*** = <nothing>
Formatted Example: #K%00005003AAD

#define *CL_SEND_UPTIME* **AAF**

This is a single data point literal.
This literal is identical to the *CL_SEND_REBOOT* literal except that we're sending the ***timestring*** as the amount of time the mud has been up instead of when it'll reboot.
Note: If the Caption (**CAP**) literal is used, the Uptime literal is not used.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_UPTIME+**timestring***
Example: ***timestring*** = 05:30
Formatted Example: #K%00005008AAF05:30

#define *CL_SEND_AVI* **AAG**

This is a multiple data point literal.
This literal is similar to the CL_SEND_IMAGE except that it is used to send the ***filename*** of the AVI movie to be displayed in the Portal© Imagery. Only AVI files are supported. You do not need to include the file extension unless you wish. All AVI's are stored in the media/images/ directory. The ***imagelabel*** (which will appear below the image in the Imagery) is optional. You can specify the ***height*** and ***width*** of the AVI also. These are optional as well. You can also choose to ***repeat*** the MIDI or not. The MIDI will repeat if ***repeat*** is anything but an empty string.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_AVI+**filename**+CL_DELIM+*
*   ***imagelabel**+CL_DELIM+**height**+CL_DELIM+**width***

(non-repeating example)
Example: ***filename*** = raging_orc.avi
Example: ***imagelabel*** = Raging Orc
Example: ***height*** = 300
Example: ***width*** = 250
Example: ***repeat*** = <nothing>
Formatted Example: #K%00005036AAGraging_orc.avi~Raging Orc~300~250

(repeating example)
Example: ***filename*** = raging_orc.avi
Example: ***imagelabel*** = Raging Orc
Example: ***height*** = 300
Example: ***width*** = 250
Example: ***repeat*** = 1
Formatted Example: #K%00005038AAGraging_orc.avi~Raging Orc~300~250~1

#define *CL_DOWNLOAD_MEDIA* **AAH**

This is a multiple data point literal.
This literal allows you to automatically download media files to the user's media\sounds and media\images directories. The available files for download must be of the kind .WAV, .MID, .RMI, .MP3 (sounds) .BMP, .GIF or .AVI (images). It requires that you specify a ***filename***, which will be the filename that will appear in the user's applicable media directory. You also must supply the ***URL*** where the media file resides on the web (usually on your MUD's site somewhere).
Notes:

- If the ***filename*** already exists in the user's applicable media directory, this operation aborts. Basically it's a good idea to keep your files named uniquely, maybe prepended with your MUD's name (e.g. blahmud_bigroar.wav).
- The ***filename*** cannot contain spaces and is limited to 255 characters in length.
- The ***filename*** must contain the file type extension (.GIF, .AVI etc.)
- The ***filename*** is not case sensitive, but the ***URL*** is.
- The ***URL*** is case sensitive, in case you missed it.
- This function is only enabled if the user has not disabled auto downloading.
- The user will be notified of the ***filename*** that was downloaded to their applicable media directory, as well as the ***URL*** from which it was downloaded.
- Try to keep the files relatively small (under 100K) as downloading them sucks extra bandwidth from the user. Also, spread them out if you can. Having the user download 100 files one right after the other is not always a good idea. A good possible usage would be a command to use on your MUD which would download a list of specific media for the user. This way they could enter the command and come back in a few minutes when it's downloaded. Portal© won't attempt to download files they already have, so it's ok to send files they don't have.
- Unless you ***KNOW*** your users have a killer connection, don't even think about sending them .MP3 files.

Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_DOWNLOAD_SOUND+**filename**+CL_DELIM+**URL***
(MIDI example)
Example: ***filename*** = B5.rmi
Example: ***URL*** = http://www.gameaxle.com/B5.rmi
Formatted Example: #K%00005040AAHB5.rmi~http://www.gameaxle.com/B5.rmi

(WAV example)
Example: ***filename*** = lock.wav
Example: ***URL*** = http://www.gameaxle.com/lock.wav
Formatted Example: #K%00005044AAHlock.wav~http://www.gameaxle.com/lock.wav

#define *CL_SEND_SPECIAL* **BAA**

This is a single data point literal.
This literal is used to send a special information ***textstring*** to Portal©. Generally this means weapon strikes, special armor messages, or other item/combat related information to send to the client. Guilds should NOT use this code. Instead, they should use the *CL_SEND_SPECIAL2* literal as detailed below.
Note: The text sent with this literal and with the *CL_SEND_SPECIAL2* literal should be 60 characters or less. This is not a hard limit, but it's the most text that can be displayed in the Portal© field without the user expanding it. (don't hack lines just to meet this limit, but try.)
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_SPECIAL+**textstring***
Example: ***textstring*** = Your sword explodes upon the goblin with a fireball!
Formatted Example: #K%00005055BAAYour sword explodes upon the goblin with a fireball!

#define *CL_SEND_SPECIAL2* **BAC**

This is a single data point literal.
This literal is used to send a special guild information ***textstring*** to Portal©, such as when a special power has kicked in, or when another has failed, or when somebody advances in guild level etc.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_SPECIAL2+**textstring***
Example: ***textstring*** = Your magical barrier of protection has faded!
Formatted Example: #K%00005048BACYour magical barrier of protection has faded!

#define *CL_SEND_TELL* **BAB**

This is a multiple data point literal.
This literal requires exactly 3 data points.
This literal is used to send 'tell' information to Portal©. The first argument is a ***direction*** indicator if the tell is originating from the player or coming to the player (i.e. 'You tell Soandso:' vs "Soandso tells you:'). The indicator is either an "x" (lower case) or an empty string. The second argument is the ***object*** of the tell (to or from this person). The third argument is the actual tell ***text***.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_TELL+**direction**+ CL_DELIM+**object**+ CL_DELIM +**text***

Here are two examples of *CL_SEND_TELL* lines:

x~Fred~Hey fred, how's it going?
~Fred~I'm doing fine, thanks for asking.

The first line indicates that I have sent a tell to Fred from me (the placement of the **x** tells us this). The second line indicates a tell coming from Fred to me. Both of these would appear on my Tell Monitor.

Formatted Example: #K%00005035BABx~Fred~Hey Fred, how's it going?
Formatted Example: #K%00005040BAB~Fred~I'm doing fine, thanks for asking.

#define *CL_SEND_ROOM* **BAD**

This is a single data point literal.
This literal is used to send the short ***description*** of the room that the player is currently in. This information is displayed at the top of the Portal© screen for quick reference.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_ROOM+**description***
Example: ***description*** = You are standing in a dark forest
Formatted Example: #K%00005036BADYou are standing in a dark forest

#define *CL_SEND_MUDLAG* **BAE**

This is a single data point literal.
If your mud supports any form of ***meter*** to display MUD-side lag, you can send it to the client via this literal. This shows in the title bar of Portal© with the uptime and reboot information.
Note: If the Caption (**CAP**) literal is used, the MUDLag literal is not used.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_MUDLAG+**meter***
Example: ***meter*** = 0.05
Formatted Example: #K%00005007BAE0.05

#define *CL_SEND_EDIT* **BAF**

This is a single data point literal.
This literal is used to send the current ***file*** that is being edited by the user/wizard. When done editing the file, send the *CL_SEND_EDIT* with an empty string. This will clear the file from the display.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_EDIT+**file***
Example: ***file*** = /obj/monster.c
Format Example: #K%00005017BAF/obj/monster.c

#define *CL_GP1_ MASK* **BBA**

This is a single data point literal.
This literal is used to send the customized ***mask*** that will be used on the hints for the Guild Point 1 controls (on the HP bars and Status Monitor, etc.) The default is "Guild Points 1" and most likely will want to be changed by your MUD or guild/class/clan. This will change it automatically.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_GP1_MASK+**mask***
Example: ***mask*** = Stamina

Format Example: #K%00005010BBAStamina

#define *CL_GP2_ MASK* **BBB**

This is a single data point literal.
This literal is used to send the customized ***mask*** that will be used on the hints for the Guild Point 2 controls (on the HP bars and Status Monitor, etc.) The default is "Guild Points 2" and most likely will want to be changed by your MUD or guild/class/clan. This will change it automatically.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_GP2_MASK+**mask***
Example: ***mask*** = Violet Plasma
Format Example: #K%00005016BBBViolet Plasma

#define *CL_HP_ MASK* **BBC**

This is a single data point literal.
This literal is used to send the customized ***mask*** that will be used on the hints for the Hit Point controls (on the HP bars and Status Monitor, etc.) The default is "Hit Points" and might wish to be changed by your MUD. This will change it automatically.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_HP_MASK+**mask***
Example: ***mask*** = Health Status
Format Example: #K%00005016BBCHealth Status

#define *CL_SP_ MASK* **BBD**

This is a single data point literal.
This literal is used to send the customized ***mask*** that will be used on the hints for the Spell Point controls (on the HP bars and Status Monitor, etc.) The default is "Spell Points" and might wish to be changed by your MUD. This will change it automatically.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SP_MASK+**mask***
Example: ***mask*** = Mana
Format Example: #K%00005007BBDMana

#define *CL_SEND_CAPTION* **CAP**

This is a single data point literal.
This literal allows you to set the user's Portal© window ***caption*** (the very top bar of the Main Screen). The string you send will appear after the standard Portal© caption elements: "<MUD> – <Player> (<online time>)" Space is already appended after the (<online time>) element, so prepending your ***caption*** with a space is not necessary. Use this literal for whatever kind of string you want to send: news flashes, player status, MUD status, etc.
Note: If this literal is used, the Reboot (**AAC**) Uptime (**AAF**) and MUDLag (**BAE**) literals are not used.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_CAPTION+**caption***
Example: ***caption*** = Newsbrief: Canada has invaded Michigan!
Formatted Example: #K%00005042CAPNewsbrief: Canada has invaded Michigan!

#define *CL_SEND_BEGIN_FILE* **CDF**

This is a multiple data point literal.
This literal requires exactly 2 data points.
This literal is used to begin a text ***file*** 'download' from the mud. The two data points are the # of ***lines*** being sent and the 'tag' to be associated with the download. This could be the file name if it's a file, or 'Who List' for instance if you're sending a who list as a file.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_BEGIN_FILE+**lines**+ CL_DELIM+**file***
Example: ***lines*** = 125
Example: ***file*** = /obj/player.c
Format Example: #K%00005020CDF125~/obj/player.c

#define *CL_SEND_CONT_FILE* **CCF**

This is a single data point literal.
This literal is used to send a ***line*** of the file or text that was initialized for download with the CL_SEND_BEGIN_FILE literal. Each line of the file will be sent with this literal.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_CONT_FILE+**line***
Example: ***line*** = if (attacker_ob) { attack(); attacked_this_round = 1; }
Format Example: #K%00005058CCFif (attacker_ob) { attack(); attacked_this_round = 1; }

#define *CL_SEND_END_FILE* **CEF**

This is a 0 data point literal.
No data is sent with this literal.
This tells Portal© that a the file being download has completed. It's a good idea to send this twice, just to ensure that Portal© ends the file transfer just in case one of them gets lost/garbled in line noise.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_END_FILE*
Format Example: #K%00005003CEF

#define *CL_SEND_CHAT* **CAA**

This is a multiple data point literal.
This literal requires exactly 4 data points.
This literal is used to send chat line information to Portal©. The 4 data points are line **command**, **line** name, **source** player name, and **data**. The Line Text should be an unformatted string (no color nor wrapping). Remember, the *CL_DELIM* literal must separate the data points.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_CHAT+**command**+*
*CL_DELIM+**line**+CL_DELIM+**source**+CL_DELIM+**data***

Example: ***command*** = guildchat
Example: ***line*** = Guild
Example: ***source*** = Rastafan
Example: ***data*** = [GUILD] Rastafan: Hi guys

This indicates that **Rastafan** has sent a chat to his guild using the **guildchat** command. The **Guild** title will be the title for this line, and the **Data** would appear on my Chat Monitor.

Formatted Example: #K%00005053CAAguildchat~Guild~Rastafan~[GUILD] Rastafan: Hi guys

#define *CL_SEND_ROOMCODE* **DDD**

This is a single data point literal.
This literal can have any number of elements within the single data point 4 data points.
This tells Portal© to display specific exits in the Room Monitor (Portal GT Upgrade Pack B and later versions). Simply include the abbreviated dirs (n, e, w, s, ne, nw, se, sw, u, d, in, out, enter and exit) separated by spaces and Portal© will display them graphically in the room monitor.
Note: "in" and "enter" are exclusive literals, since the graphical representation is identical. Whatever you send first will be used for the command. The same applies to "out" and "exit".
The Room Monitor supports up to three additional, non-standard directions that will be displayed as clickable text. Just append them to the end of the list to be used.
Note: The non-standard exits as well as in, out, enter and exit require client Version 8.8 or above.
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_ROOMCODE*
Format Example: #K%00005011DDDn e ne u
Format Example: #K%00005011DDDn e ne u in
Format Example: #K%00005011DDDn e ne u exit leave climb

This indicates that the room monitor is to display a room with exits "n" "e" "ne" and "u"

#define *CL_SEND_COMPOSITE* **FFF**

This is a multiple data point literal.
This literal can have any number of data points.
See the examples in the main text on how to create a *COMPOSITE* data line.
The following *HP_DATA_CODE* literals are valid for the *COMPOSITE* line code literal:
Format: *ACTIVATE+SEC_CODE+CHAR_COUNT+CL_SEND_COMPOSITE+* ***composite1****+CL_DELIM+****composite2****+CL_DELIM+****composite3****+…+ CL_DELIM+****compositeN***

#define *CL_SEND_HP* **A**
This literal is used to send the player's current hit points.
Range for this literal is 0-9999.
Example: 312
Formatted Example: #K%00005007FFFA312

#define *CL_SEND_MAXHP* **B**
This literal is used to send the player's max hit points.
Range for this literal is 0-9999.
Example: 410
Formatted Example: #K%00005007FFFB410

#define *CL_SEND_SP* **C**
This literal is used to send the player's current spell points.
Range for this literal is 0-9999.
Example: 300
Formatted Example: #K%00005007FFFC300

#define *CL_SEND_MAXSP* **D**
This literal is used to send the player's max spell points.
Range for this literal is 0-9999.
Example: 425
Formatted Example: #K%00005007FFFD425

#define *CL_SEND_GP1* **E**
This literal is used to send the player's current primary guild points.
Range for this literal is 0-9999.
Example: 57
Formatted Example: #K%00005006FFFE57

#define *CL_SEND_MAXGP1* **F**
This literal is used to send the player's max primary guild points.
Range for this literal is 0-9999.
Example: 502
Formatted Example: #K%00005007FFFF502

#define *CL_SEND_GP2* **G**
This literal is used to send the player's current secondary guild points.
Range for this literal is 0-9999.
Example: 50
Formatted Example: #K%00005006FFFG50

#define *CL_SEND_MAXGP2* **H**
This literal is used to send the player's max secondary guild points.
Range for this literal is 0-9999.
Example: 100
Formatted Example: #K%00005007FFFH100

#define *CL_SEND_GLINE1* **I**
This literal is used to send the player's primary guild line information.
Range for this literal is 0-60 characters.
Example: Next Level: 5000gxp Shield Strength: 100%
Formatted Example: #K%00005045FFFINext Level: 5000gxp Shield Strength: 100%

#define *CL_SEND_GLINE2* **J**
This literal is used to send the player's secondary guild line information.
Range for this literal is 0-60 characters.
Example: Guild Level: 32 Familiar Health: 75%
Formatted Example: #K%00005040FFFJGuild Level: 32 Familiar Health: 75%

#define *CL_SEND_ATTACKER* **K**
This literal is used to send the name of the monster the player is currently fighting.
Range for this literal is 0-60 characters.
Example: Shamus the Shop Keeper
Formatted Example: #K%00005026FFFKShamus the Shop Keeper
To set the value to an empty string, send the CL_DELIM (~) after the **K** code
Example: <nothing> (clears the enemy name)
Formatted Example: #K%00005025FFFK~

#define *CL_SEND_ATTCOND* **L**
This literal is used to send the % of max hit points their current enemy has.
Range for this literal is 0-9999 (> 100 means above max HP). Again, this is % HP of the enemy, not actual HP.
Example: 75
Formatted Example: #K%00005006FFFL75
Note: Sending a 0 will set the enemy name to gray on the Status Monitor and in the Imagery. Setting it to any value above 0 will set it to black. This is supported in Portal© versions 5b and above.

#define *CL_SEND_ATTIMG* **M**
This literal is used to send the image file for the monster they are fighting.
Range for this literal is restricted by Windows' file naming conventions. As a general rule, you should not use anything but alphanumeric characters (a, b, c, 1, 2, 3) and the underscore (_). The filename is not case-sensitive. The length of the total filename cannot be longer than 255 characters.
Example: townfolk_shamus
Formatted Example: #K%00005019FFFMtownfolk_shamus

You can send any number of composites from **A** to **M** via the *CL_SEND_COMPOSITE* literal. Here is an example to send ***HP***, ***SP***, ***GP1*** and attacker ***condition*** all at once.
Example: *HP* = 312
Example: *SP* = 300
Example: *GP1* = 57
Example: *condition* = 75
Formatted Example: #K%00005020FFFA312~C300~E57~L75

It is much better to send composite literals in groups this way, compared to sending them individually. You could have accomplished the above by sending four separate *CL_SEND_COMPOSITE* literals, but that would be a horrible waste of bandwidth. Sending it in a group as shown gains the same results in one efficient sweep.

Appendix B: Coding Suggestions & General Hints

Here are some tips to help on bandwidth concerns:

- For the *COMPOSITE* literal the best and easiest way to handle sending that information is to track the values from one round to the next and only send the values that have changed. This reduces the overall bandwidth and makes the send a lot smoother. Basically, don't send two *COMPOSITE* literals, one sending HP and the other HPMax. Have one *COMPOSITE* literal send both at once.
- Assuming you're doing the above method of tracking and sending, for your *CL_SEND_GLINE1* (and 2) calls, it is best to group values that change frequently into only **one** of the gline calls, and values that don't change frequently into the other. This reduces bandwidth by only sending one of the lines frequently and the other seldom. For example, put guild experience (gexp) to next level and guild age into one of them. Put guild level and alignment (or some such) in the other.
- Build a client.c router and a few simul_efuns that call it. This will make all your wizards a lot more likely to use it if it is easy to do so. Put the above #define statements into a client.h file so that all the wizards can #include it and just use the defines instead of the key codes. That way if the codes ever change there will only need to be one file changed (though files that #included the old client.h file will need to be updated/loaded).
- Create a #define *CL_DELIM* "~" for that same purpose as above.
- This may seem obvious, but get Portal© yourself and make one change at a time and test it, this could save you a lot of trouble in the long run.
- Sending images and sounds is very easy, but doesn't do you any good if the players don't have them. So post sound and image 'packs' on your website somewhere and encourage your players to download and place them into their media/images or media/sounds directories. You can't really appreciate things like this until you see/hear them. It's one thing to open a door, but it's insanely more exciting to hear the sound of a creaking door when you do so.
- Don't forget the "3klient" command, this is what turns it all on!

Appendix C: Coloring the Gline1 & Gline2

The gline literals (*CL_SEND_GLINE1* & *CL_SEND_GLINE2*) appear on the HP3 Bar. The text that appears has a black background and the font is colored white by default. While you cannot change the black background, you can change the font color.

All you have to do is encapsulate the text you desire to color in the format specified below. You can't use < or > anywhere in the glines as normal text or it'll screw up. If you already use < or > enclosures, you'll have to use something else to encapsulate your data. All text not enclosed in any color activators appear as white.

Format: <xword>
x color code
word text to be colored (any length)

Valid Color Codes

y	yellow
r	red
b	blue
g	green (windows calls it lime)
c	cyan (windows calls it aqua)
v	violet (windows calls it fuchsia)
s	silver (dim white)

Example 1: The red <rfox> jumped over the cyan <cdog>
Display: **The red fox jumped over the cyan dog**

Example 2: Spells Activated: [<bshield>,<gmagic missile>]
Display: **Spells Activated: [shield,magic missile]**

Example 3: Demon Follower's Health: <a95%>
Display: **Demon Follower's Health: 95%**

Now, why did it appear all in white? Because "a" was used, which is not a valid code.

We suggest you keep the number of uses of the activator to under 10 per gline. That means under 10 usages of the <> pairings.

Appendix D:FTP Emulation Upload

While the literals *CL_SEND_BEGIN_FILE*, *CL_SEND_CONT_FILE* and *CL_SEND_END_FILE* (defined in Appendix A) take care of the FTP Emulation for download (server to client) transfer, there is also functionality for upload (client to server) transfer. This is done mostly via an added command on the MUD in the format of "**3ksendfile** *filename*" which will be used by Portal©. **The upstream transfer can only transfer text files**. The user can select any number of files to upload via this command, but each one is buffered to be sent every 2 seconds.

Lines following the **3ksendfile** command are added to the file *filename* in the order they are received. This is not unlike sending a slew of lines into the standard "ed" editor found on most MUDs. Here the server should build up an array buffer for the lines, which will then be written to the final file *filename*.

When the transfer of lines is complete, Portal© sends the string **EOF**. When this string comes across, it tells the server that the file transfer is complete and the array buffer can then be dealt with accordingly (dumped into a file, posted to a board, etc.).

This is not recommended for use on files more than 300 lines. Standard FTP should be used for such transfers. Windows filenames will have any spaces converted to underscores upon transfer. Again, this will only work with text files.

Portal© for Windows MIP Terms of Use License

The information contained within this document (hereafter referred to as the MIP) is provided "as is." In no event shall the author(s), GameAxle, The Marble Group Inc., Giftwicks LLC or any other related parties be liable for any consequential, special, incidental or indirect damages of any kind arising out of the delivery, performance or use of the MIP.

Other than that, have at it. As of 4/1/08, Portal© is open-source, so have fun and promote the spirit of mudding to the fullest extent of your imagination!