Generic Mud Communication Protocol

MUD servers often want to send additional data to a MUD client that doesn't necessarily need to be displayed, as well as needing a way to identify clients that support out-of-band data.

The history of GMCP starts with the Achaea Telnet Client Protocol (ATCP) using TELNET code 200 which was implemented by cMUD in 2008. ATCP however only allowed for sending plain text. MSDP (Mud Server Data Protocol) was developed in 2009 and provides a standardized way to define typeless variables, arrays, tables, and commands. In 2010 IRE launched mudstandards.org in an attempt to update the ATCP protocol with input from the wider MUD community.

This resulted in the conceptualization of ATCP2 using TELNET code 201, which was later renamed to GMCP. GMCP uses the JSON syntax to define structured and typed data.

At some point mudstandards announced they were unhappy with the community effort and would privately develop GMCP with CMUD, where GMCP support and an interface would be hardwired into CMUD. In response Aardwolf MUD went ahead and implemented GMCP independently, and TinTin++ and MUSHclient provided working interface scripts for Aardwolf a few weeks later. The out-of-the-box CMUD interface never materialized.

As mudstandards.org became defunct in 2011 this document provides a technical description of the GMCP protocol as well as MSDP over GMCP. MSDP over GMCP offers standardized generic event handling besides sending structured data. Servers that implement MSDP over GMCP will be able to use both the MSDP and JSON standard for defining structured data as well as perform MSDP event handling in either format.

The GMCP Protocol

GMCP is implemented as a Telnet option **RFC854**, **RFC855**. The server and client negotiate the use of GMCP as they would any other telnet option. Once agreement has been reached on the use of the option, option sub-negotiation is used to exchange information between the server and client.

Server Commands

IAC WILL GMCP	Indicates the server wants to enable GMCP.
IAC WONT GMCP	Indicates the server wants to disable GMCP.

Client Commands

IAC DO GMCP	Indicates the client accepts GMCP sub-negotiations.
IAC DONT GMCP	Indicates the client refuses GMCP sub-negotiations.

Handshake

When a client connects to a GMCP enabled server the server should send IAC WILL GMCP. The client should respond with either IAC DO GMCP or IAC DONT GMCP. Once the server receives IAC DO GMCP both the client and the server can send GMCP sub-negotiations.

The client should never initiate a negotiation, if this happens however the server should abide by the state change. To avoid trigger loops the server should not respond to negotiations from the client, unless it correctly implements the Q method in **RFC 1143**.

Disabling GMCP

When a typical MUD server performs a copyover it loses all previously exchanged GMCP data. If this is the case, before the actual copyover, the MUD server should send IAC WONT GMCP, the client in turn should fully disable GMCP. After the copyover has finished the server and client behave as if the client has just connected, so the server should send IAC WILL GMCP.

When a typical MUD client loses its link and reconnects it loses all previously exchanged GMCP data. The server should reset its GMCP state and re-negotiate GMCP whenever a client reconnects.

GMCP definitions

GMCP	201

Example MSDP over GMCP handshake

server - IAC WILL GMCP

client - IAC DO GMCP

client - IAC SB GMCP 'MSDP {"LIST" : "COMMANDS"}' IAC SE

server - IAC SB GMCP 'MSDP {"COMMANDS":

```
["LIST","REPORT","RESET","SEND","UNREPORT"]}' IAC SE
```

The single quote characters mean that the encased text is a string, the single quotes themselves should not be send.

Data Format

Each GMCP data exchange should use the IAC SB GMCP <package.subpackage.command> <data> IAC SE format.

The package name can be case insensitive. When using MSDP over GMCP the package name is considered case sensitive and MSDP must be fully capitalized. There are no subpackages as these are not necessary.

The <data> field is optional and should be separated from the package field with a space. When sending a command without a data section the space should be omitted. The data field must use the JSON data syntax with keywords being case sensitive using UTF-8 encoding.

Packages

Each MUD server is expected to define and document its own packages. The document you are currently reading is primarily focused on describing the MSDP over GMCP functionality built into TinTin++ 2.02.0 and MTH 1.5.

MSDP over GMCP

Writing and documenting GMCP packages can be quite the hassle. MSDP is self-documenting and relatively easy to implement because it comes with two public domain flexible event handlers written in C.

MSDP over GMCP combines the popularity of JSON with the versatile command system of MSDP. It also allows the client to choose between using either MSDP, GMCP, or both. While JSON is easier for humans to read MSDP is easier for machines to read, which makes MSDP preferable for creating dynamically generated tables. In the case a client enables both MSDP and GMCP the client is expected to be able to process both MSDP and GMCP data interchangably, similarly the server is expected to process both MSDP and GMCP data interchangably, as is the case in MTH 1.5.

Keep in mind that JSON does not allow sending control-codes while MSDP does.

As things currently stand MSDP over GMCP is primarily used for client to client communication by TinTin++, offering scripters a more readable and standardized data format compared to MSDP, with a minimal implementation and maintenance burden. MSDP over GMCP might also be useful for MUDs that support MSDP and want to offer a workaround for clients that only support GMCP.

The MSDP Protocol Definition.

ATCP2

Currently Achaea, Aardwolf, MUME, Avatar, Gensis, and MUSHclient provide package definitions modeled after the ATCP2 draft. It's not the intention of this document to add to this growing list.

Discussion

MSDP Discussion Reddit board

MSDP Discussion Discord channel

Links

If you have any comments you can email me at mudclient@gmail.com.

Clients supporting both MSDP and GMCP

Axmud

Mudlet (doesn't allow control codes over MSDP)

TinTin++

Codebases

BasedMUD - Full MSDP support using MTH 1.5. Based on DikuMUD / ROM / QuickMUD / BaseMUD.

Lowlands - Full MSDP over GMCP support. Based on DikuMUD / MrMud.

NekkidMUD - Full MSDP over GMCP support. Based on SocketMUD / NakedMud.

TinTin++ - Full MSDP over GMCP support. Offers a variable set geared towards client 2 client communication.

WickedMUD - Full MSDP over GMCP support. Based on SocketMUD.

Discussion

MSDP Discussion Reddit board

MSDP Discussion Discord channel

Servers

sotmud.slackhalla.org:6969

Snippets

Scandum's MUD Telopt Handler - Handles CHARSET, EOR, MCCP2, MCCP3, MSDP, MSDP over GMCP, MSSP, MTTS, NAWS, NEW-ENVIRON, TTYPE, and xterm 256 colors.