GMCP Specification Document

TABLE OF CONTENTS

1 Module Support

Module and message names are not case-sensitive. JSON key names are case-sensitive.

1.1 Supported Module List

Module Name	Purpose
Core	Core functionality
Char	Information about a character
Char.Skills	Information about skills known by the player
Char.Items	Information about items in a player's inventory and room
Comm.Channel	Identification of communication channels and players on the channel
Room	Various information about the current room
Redirect	Redirect output to another window
IRE.Composer	IRE-specific, used for editing bigger texts client-side
IRE.Display	Used by the HTML 5 client
IRE.FileStore	Used internally by the IRE Clients
IRE.Misc	Used internally by the IRE Clients
IRE.Rift	IRE-specific, transmits information about a player's Rift contents
IRE.Sound	IRE-specific, support for playing sound effects in the HTML5 client only
IRE.Tasks	Information about player tasks
IRE.Time	IRE-specific, used for sending the current Achaean date and time
IRE.Wiz	Used internally by the IRE Clients

1.2 Supported Commands Per Module

Module Name	Sent By Client	Sent By Server
Core	Core.Hello Core.Supports.Set Core.Supports.Add Core.Supports.Remove Core.KeepAlive Core.Ping	Core.Ping Core.Goodbye
Char	Char.Login	Char.Status Char.StatusVars Char.Vitals
Char.Skills	Char.Skills.Get	Char.Skills.Groups Char.Skills.Info Char.Skills.List

Char.Items	Char.Items.Contents Char.Items.Inv	Char.Items.Add Char.Items.List Char.Items.Remove Char.Items.Update
Comm.Channel	Comm.Channel.Players	Comm.Channel.End Comm.Channel.List Comm.Channel.Players Comm.Channel.Start Comm.Channel.Text
Room		Room.AddPlayer Room.Info Room.Players Room.RemovePlayer Room.WrongDir
Redirect		Redirect.Window
IRE.Composer	IRE.Composer.SetBuffer	IRE.Composer.Edit
IRE.Rift	IRE.Rift.Request	IRE.Rift.Change IRE.Rift.List
IRE.Tasks	IRE.Tasks.Request	IRE.Tasks.List IRE.Tasks.Completed
IRE.Time	IRE.Time.Request	IRE.Time.List IRE.Time.Update

2 MODULE DETAILS

2.1 CORE

Sent by client:

- Core.Hello
  - Needs to be the first message that the client sends, used to identify the client.
  - Message body is an object with the keys "client" and "version" containing the client's name and version
    e.g. Core.Hello { "client": "Nexus", "version": "3.1.90" }

- Core.Supports.Set
  - Notifies the server about what GMCP modules are supported by the client.
  - If another Core.Supports.* message has been received earlier, the list is deleted and replaced with this new one.
  - Message body is an array of strings, each consisting of the module name and whether it is enabled, separated by a space.
  - Most client implementations will only need to send Set once and won't need Add/Remove; exceptions are module implementations provided by plug-ins.
    e.g. Core.Supports.Set ["Char 1", "Char.Skills 1", Char.Items 1"]

- Core.Supports.Add
  - Similar to .Set but appends the supported module list to the one sent by an earlier .Set message.
  - If no .Set message has been sent yet, the behaviour of .Add is identical to that of .Set
  - If the list includes module names that were already included earlier, information sent in the .Add takes precedence.
  - Message body format is identical to that of .Set.

- Core.Supports.Remove
  - Removes the specified modules from the list of supported modules.
  - Message body format is similar to Set, but any appended 1 or 0 is ignored.
    e.g. Core.Supports.Remove ["Char", "Char.Skills", "Char.Items"]

- Core.KeepAlive
  - Causes the server to reset the timeout for the logged in character. Has no body.

- Core.Ping
  - Causes the server to send a Core.Ping message back
  - Message body is a number which indicates average ping time from previous requests, if available.
    e.g. Core.Ping 120

Sent by server:

- Core.Ping
  - Sent in reply to client-initiated Core.Ping. Has no body.

- Core.Goodbye
  - Sent by server immediately before terminating a connection.
  - Message body is a string to be shown to the user – it can explain the reason for the disconnect.
    e.g. Core.Goodbye "Goodbye, adventurer"

2.2 CHAR

Sent by client:

- Char.Login
  - Used to log in a character, only interpreted if no character is logged in for that connection
  - Message body is an object with keys "name" and "password"
    e.g. Char.Login { "name": "somename", "password": "somepassword" }

Sent by server:

- Char.Vitals
  - Basic character attributes such as health, mana, etc.
  - Message body is an object containing several variables
  - Additionally, each variable is included in a string, in the format name:current/max
  - Interpretation of the variables is game-specific.
  - It is generally safe to assume that the values are numbers (even though encoded as strings).
    e.g. Char.Vitals { "hp": "4500", "maxhp": "4800", "mp": "1200", "maxmp": "2500", "ep": "15000", "maxep": "16000", "wp": "14000", "maxwp": "15000", "nl": "10", "string": "H:4500/4800 M:1200/2500 E:15000/16000 W:14000/15000 NL:10/100" }

- Char.StatusVars
  - Sent by the server after a successful login or after the module is enabled.
  - Contains a list of character variables (level, race, etc).
  - Message body is an object where each contained element is a name-caption pair, name is the internal name and caption the user-visible one.
    e.g. Char.StatusVars { "level": "Level", "race": "Race", "house": "House" }

- Char.Status
  - Values of character variables defined by .StatusVars
  - A full list is sent by the server right after StatusVars and changes are sent in subsequent messages as they occur.
  - With the exception of the initial Status message, messages only contain changed values; if a variable is not included, it has not changed since the previous Status message
  - Message body is an object where contained elements are name-value pairs, name is the internal name defined by the StatusVars message and value is the variable value.
    e.g. Char.Status { "level": "58", "city": "Mhaldor" }

2.3 CHAR.SKILLS

Sent by client:

- Char.Skills.Get
  - Sent by client to request skill information
  - Message body is an object with keys "group" and "name"
  - If both group and name is provided, the server will send Char.Skills.Info for the specified skill
  - If group is provided but name is not, the server will send Char.Skills.List for that group otherwise the server will send Char.Skills.Groups
    e.g. Char.Skills.Get { "group": "elementalism", "name": "firelash" }

Sent by server:

- Char.Skills.Groups
  - Groups of skills available to the character
  - Sent by server on request or at any time (usually if the list changes)
  - For IRE games, groups are skills like Survival or Elementalism
  - message body is an array of strings, each being one name
    e.g. Char.Skills.Groups ["Survival", "Enchantment", "Elementalism", "Crystalism"]

- Char.Skills.List
  - List of skills in a group available to the character
  - Sent by server on request only
  - For IRE games, this is the list visible on AB <skillname>
  - Message body is an object with keys "group" and "list", where group is the group name as a string
  - The list value is an array of strings, each being the name of one skill
    e.g. { "group": "Elementalism", "list": ["Channel", "Light", "Gust"] }

- Char.Skills.Info
  - Information about a single skill, only sent upon request
  - Message body is an object, keys are "group", "skill", and "info", values are strings
  - Group and skill identify the request, info is a description (usually multi-line) of the skill's functionality and usage
    e.g. Char.Skills.Info { "group": "Elementalism", "skill": "Firelash", "blah blah" }

2.4 CHAR.ITEMS

Sent by client:

- Char.Items.Contents
  - Sent by client to request the contents of an item in the player's inventory
  - Message body is a number identifying the item, causes the server to send back an appropriate Char.Items.List message.'
    e.g. Char.Item.Contents 12345

- Char.Items.Inv
  - Sent by client to request a list of the items in the player's inventory
  - Message body is empty, causes the server to send back an appropriate Char.Items.List message
    e.g. Char.Items.Inv ""

Sent by server:

- Char.Items.Add
  - Informs the client about an item being added to the specified location
  - Message body is an object with keys "location" and "item"
  - Location is same as with List, item is an object with the same structure as one from the items array of List
    E.g. Char.Items.Add { "location": "room", "item": { "id": "60572", "name": "an ornate steel rapier" } }

- Char.Items.List
  - List of items at a specified location (room, inv, held container)
  - Message body is an object with keys "location" and "item"
    E.g. Char.Items.List { "location": "inv", "items": [{ "id": "26545", "name": "a red ink", "attrib": "gr" }, { "id": "60572", "name": "an ornate steel rapier" }] }

- Char.Items.Remove
  - Removes an item from the list of items in the player's inventory or current room
    E.g. Char.Items.Remove { "location": "inv", "item": { "id": "60572", "name": "an ornate steel rapier" } }

- Char.Items.Update
  - Updates the details about an item in the players inventory or the current room.
    E.g. Char.Items.Update { "location": "inv", "item": { "id": "60572", "name": "an ornate steel rapier" } }

2.5 COMM.CHANNEL

Sent by client:

- Comm.Channel.Players
  - Sent by client to request a list of all visible players and the channels that are shared.
    e.g. Comm.Channel.Players ""

Sent by server:

- Comm.Channel.End (depreciated)
  - Indicates the end of communication on a channel you can hear
    E.g. Comm.Channel.End "says"

- Comm.Channel.List
  - A listing of all the communication channels available to a player including the name, the command to access it, and any caption that appears on it.
    E.g. Comm.Channel.List [{ "name": "newbie", "caption": "Newbie", "command": "newbie" }, { "name": "market", "caption": "Market", "command": "market" }]

- Comm.Channel.Players
  - list of players and organizations (city, house, ...) that they share with this player
  - message body is an array with each element describing one player
  - each element is an object with keys "name" and "channels", name is a string, channels is an array
  - the channels array may be omitted if empty; if given, it is a list of organization names
    E.g. Comm.Channel.Players [{"name": "Player1", "channels: ["Some city", "Some house"]}, {"name": "Player2"}]
  -

- Comm.Channel.Start (depreciated)
  - Indicates the start of communication on a channel you can hear.
    E.g. Comm.Channel.Start "says"
- Comm.Channel.Text
  - The text of the communication that you heard, who spoke, and which channel it was on
    E.g. Comm.Channel.Text { "channel": "says", "talker": "Tecton", "text": "(Tecton the Terraformer says, \"Are we releasing dragon lairs or the phase artefact first?\"" }

2.6 Room

Sent by server:

- Room.Info
  - Contains information about the room that the player is in. Some of these may be IRE-specific
  - Message body is an object with the following keys
    - "num" - number identifying the room
    - "name" - string containing the brief description
    - "area" - string containing the area name
    - "environment" - string containing environment type ("Hills", "Ocean", …)
    - "coords" - room coordinates (string of numbers separated by commas – area,X,Y,X,building – building is optional)
    - "map" - map information - URL pointing to a map image, followed by X and Y room (not pixel) coordinates on the map
    - "details" - array holding further information about the room - shop, bank,…
    - "exits" - object containing exits, each key is a direction and each value is the number identifying the target room

    E.g. Room.Info {"num": 12345, "name": "On a hill", "area": "Barren hills", "environment": "Hills", "coords": "45,5,4,3", "map": "www.achaea.com/irex/maps/clientmap.php?map=45&level=3 5 4", "exits": { "n": 12344, "se": 12336 }, "details": ["shop", "bank"] }

- Room.WrongDir
  - Sent if the player tries to move in a non-existant direction using the standard movement commands. Upon receiving this message, the client can safely assume that the specified direction does not lead anywhere at this time
  - Message body is a string with the name of the non-existant exit
    E.g. Room.WrongDir "ne"

- Room.Players
  - Object containing player details, each key is the short name of the player and each value is the full name including titles for the player
    E.g. Room.Players {"tecton":"Tecton, the Terraformer", "cardan":"Cardan, the Curious"}

- Room.AddPlayer
  - Message body has the same object structure as Room.Players except that it only contains the one player being added to the room.
    e.g. Room.AddPlayer { "name": "Cardan", "fullname": "(Cardan, the Curious)" }

- Room.RemovePlayer
  - Message body has the same object structure as Room.Players except that it only contains the one player being removed from the room.
    e.g. Room.RemovePlayer "Cardan"

2.7 REDIRECT

Sent by server:

- Redirect.Window
  - Specifies a window to redirect further input to
  - Message body is a string specifying the window to redirect to
  - The main window is referred to as "main", and is the default if the message body is omitted or empty.
    E.g. Redirect.Window "map"

2.8 IRE.COMPOSER

Sent by client:

- IRE.Composer.SetBuffer
  - Sent by the client upon successfully editing a text which was sent to the client in an IRE.Composer.Edit message earlier
  - Sending this message only changes the edit buffer and does not end the editing session
  - On IRE games, the client may send the command *save to save a text, or the command *quit to abort editing (IRE.Composer.SetBuffer is not sent in this case) - this behaviour is IRE-specific and is one of the reasons why the Composer module is in the IRE namespace
    e.g. IRE.Composer.SetBuffer "Some written text."

Sent by server:

- IRE.Composer.Edit
  - sent by the server when the player enters an in-game editor. Body is an object, with keys "title" and "text". Text contains the current buffer, title is a title that can be shown to the user
    e.g. IRE.Composer.Edit { "title": "Composer", "text": "" }

2.9 IRE.RIFT

Sent by client:

- IRE.Rift.Request
  - Sent by client to request the server send the contents of the player rift in an IRE.Rift.List message. Request message has no body.
    e.g. IRE.Rift.Request

Sent by server:

- IRE.Rift.List
  - contents of a Rift storage
  - sent upon receiving the IRE.Rift.Request message
  - message body is an array, with each element being an object containing three keys
    - "name" is item name
    - "amount" is a number holding the item's amount, and
    - "desc" is user-visible description

    E.g. IRE.Rift.List [{ "name": "torus", "amount": "2", "desc": "crystal torus" }, { "name": "sphere", "amount": "2", "desc": "crystal sphere" }, { "name": "iron", "amount": "20", "desc": "iron" }]
- IRE.Rift.Change
  - sent whenever the item amount in a Rift changes
  - message body is an object with the same structure as one element of an array sent with the IRE.Rift.List message
    E.g. IRE.Rift.Change { "name": "silver", "amount": "13", "desc": "silver bar" }

2.10 IRE.TASKS

Sent by client:

- IRE.Tasks.Request
  - Sent by client to request tasks information. Message has no body.
    E.g. IRE.Tasks.Request

Sent by server:

- IRE.Tasks.List
  - List of players tasks, both completed and active.
  - message body is an array, with each element being an object containing seven keys.
    - "id" - numeric id for task
    - "name" - title of task
    - "desc" - verbose description of task
    - "type" - Type of task.
    - "cmd" - The command used to complete the task
    - "status" - boolean of whether task has been completed or not
    - "group" - Group task is organised into. All tasks are moved to the 'Completed' group once completed.

    E.g. IRE.Tasks.List [{ "id": "1", "name": "Break Free of Your Imprisonment", "desc": "Where are you? What's going on? There's no time to waste, you need to get out of here!\n\nPay close attention to the directions and tips on your screen and you'll be out of the dungeon in no time.", "type": "Task", "cmd": "TASK 1 INFO", "status": "1", "group": "Completed" }, { "id": "8", "name": "Using TASKS", "desc": "To help you learn all the essential commands to play Achaea, you have been given a checklist of things to do. You can view them all by typing \"TASKS\". To see the tasks you have already finished, use \"TASKS COMPLETED\".\n\nYou can do these in any order you like, but don't forget the mission your friends gave you to investigate Beku! To get started, take a look at \"TASK 12\".", "type": "Task", "cmd": "TASK 8 INFO", "status": "1", "group": "Completed" }, { "id": "30", "name": "Greet the Pixie Queen or the Imp Lord", "desc": "In addition to the pygmies, several other groups of denizens live in the land of Minia. Many of them need help and will gladly send you on quests, which reward you with gold and experience! \"HELP NEWBIE QUESTS\" lists many such quests!\n\nFor this task, travel to the pixie village, find the pixie queen, and \"GREET QUEEN\" to see what she needs. You can also go to the Ember Tower, find the imp lord, and \"GREET LORD\" to help the imps instead.", "type": "Task", "cmd": "TASK 30 INFO", "status": "0", "group": "Lending a Hand" }]

- IRE.Tasks.Completed (not used presently)

2.11 IRE.TIME

Sent by client:

- IRE.Time.Request
  - Sent by client to request time information. Message has no body.
    E.g. IRE.Time.Request

Sent by server:

- IRE.Time.List
  - List of time details as current in game.
    E.g. IRE.Time.List { "day": "3", "mon": "7", "month": "Valnuary", "year": "649", "hour": "46", "time": "It is dusk in Achaea.", "moonphase": "Waxing Crescent", "daynight": "103" }

- IRE.Time.Update
  - Sends messages to provide updates to the current In-Game time
    E.g. IRE.Time.Update { "daynight": "75" }