

# Mudlet GMCP tutorial

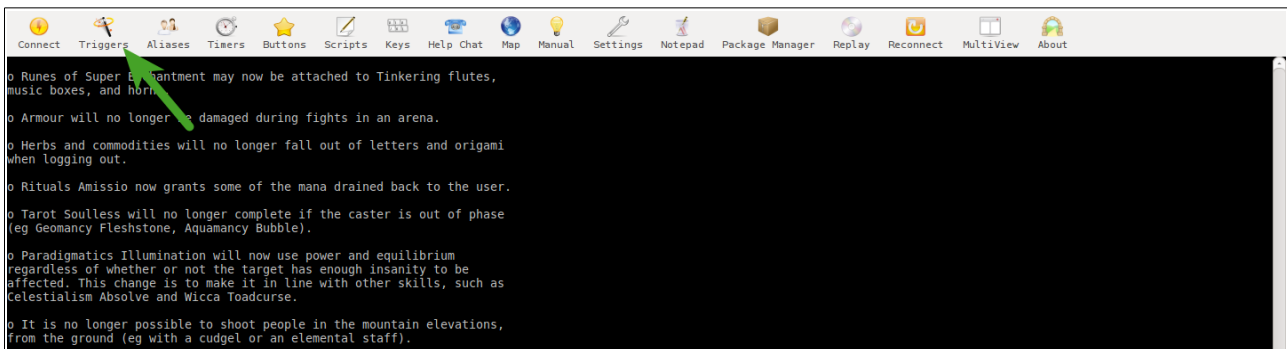
Welcome to the wonderful, invisible world of GMCP that'll make your life a lot easier! GMCP is a data channel that's implemented in IRE and other MUDs that provides you with easier access to many types of new info - like your vitals stats, detailed geographical data as to where you are, what items you have, and what skills you've got!

So let's get started with it, then. This will be a thorough, newbie-friendly tutorial filled with pictures you can spy at - and we'll be using the latest version of the freely-available MUD client, on Windows, Mac and Linux OS, called Mudlet ([get it here](#)), which supports GMCP. Once you've got it, make sure that you've got GMCP enabled (Settings → General), and you're all set to start learning!

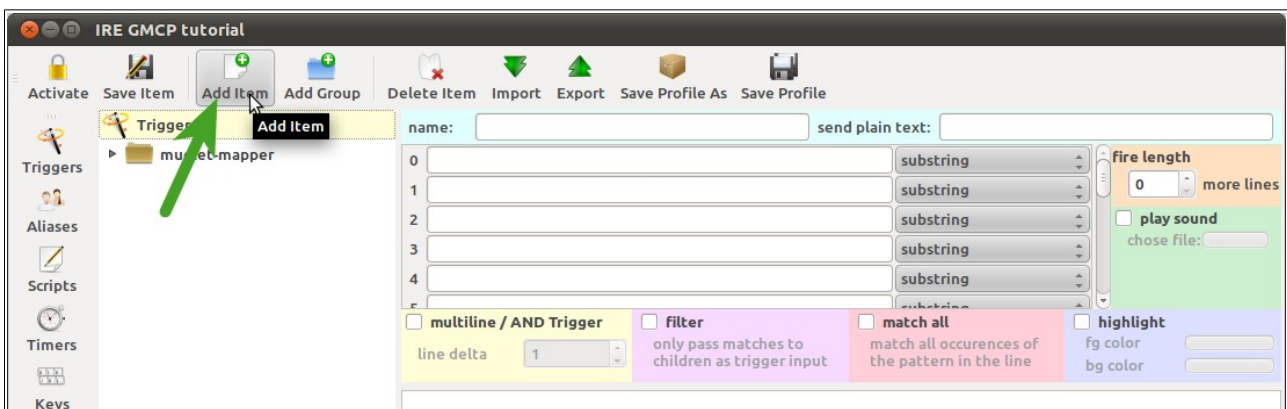
## Retrieving GMCP data

One of the most useful GMCP features across all IRE games is that with each prompt comes easily-accessible character vitals (like health, mana) data. This removes the need to make complicated [regex](#) patterns that work with all different kinds of prompts - this solution will take care of that! So **let's make a prompt trigger that will easily capture our vitals for us.**

Start off by clicking on *Triggers*:



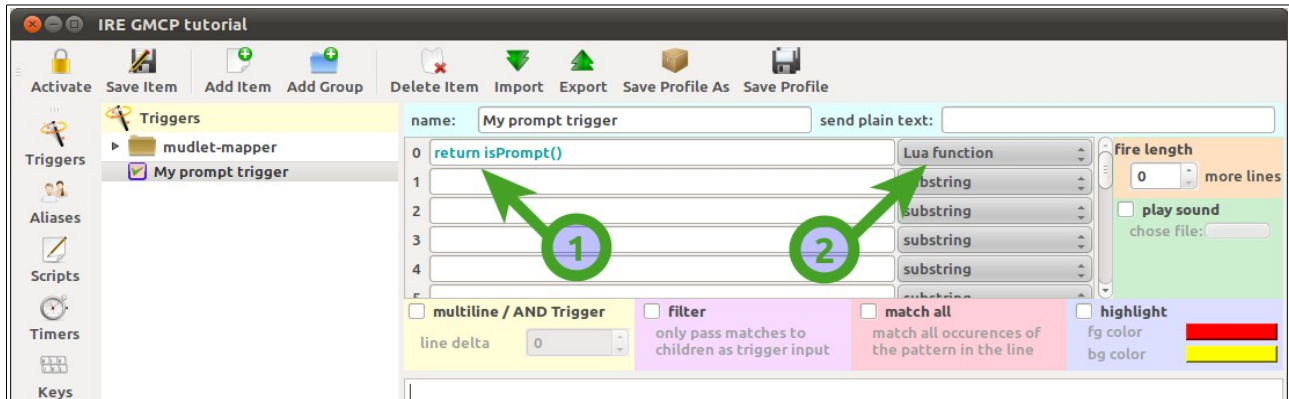
Then *Add Item*:



Add

```
return isPrompt()
```

to the first field, and by setting the pattern type to *Lua function*



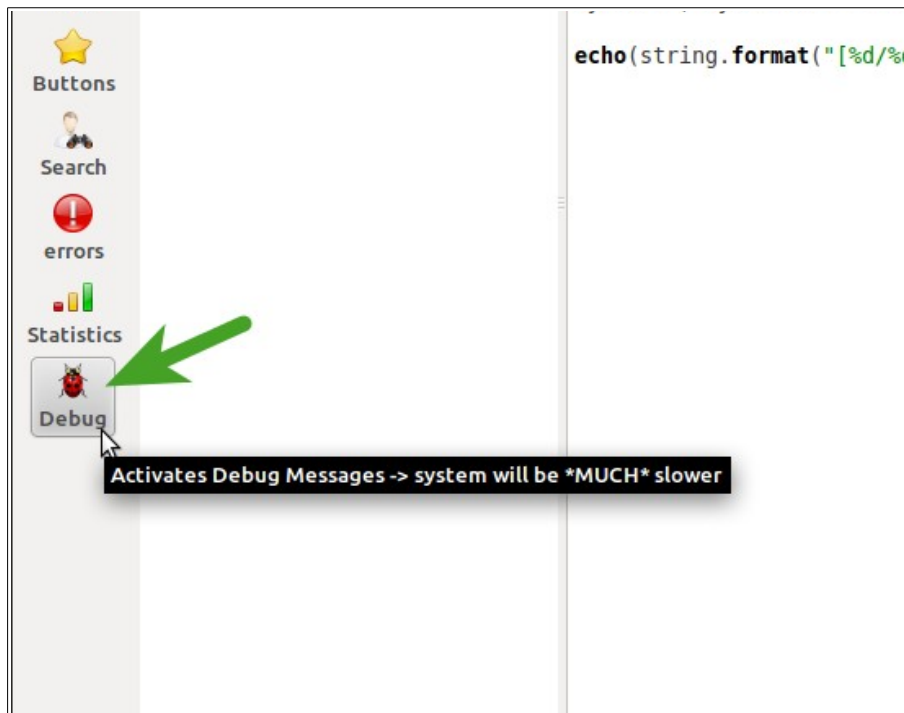
We've got ourselves a basic prompt trigger now (*\* - isPrompt()* will only work on MUDs that **do not** say <No GA> bottom-left window). Let's add GMCP magic, and use it to capture our current and max health/mana amounts into variables *myhealth* and *mymaxhealth*. Add the code from the box below into the big white space in your new trigger:

```
myhealth, mymaxhealth = tonumber(gmcp.Char.Vitals.hp), tonumber(gmcp.Char.Vitals.maxhp)
echo(string.format("[%d/%dh]", myhealth, mymaxhealth))
```

Should everything be setup alright, our echo will be doing this on every prompt, verifying that our data capture is working:

```
Deadly gossamer forest.
The loud, echoing caw of crows permeates the air, filling the area with a feeling of dread from a
war shrine of Viravain nearby. A hemlock sapling clings tenaciously to the ground here. Shadows
flock about the imposing figure of this Daughter of Shadow, swarming every inch of her body in their
unforgiving grasp. An unnatural gloom surrounds a Daughter of Darkness, filling the air with an
oppressive silence.
You see exits leading east and northwest.
5225h, 7590m, 6600e, 10p, 32400w elrx<--[5225/5225h]nw
Fetid vale of a lone pine.
The loud, echoing caw of crows permeates the air, filling the area with a feeling of dread from a
war shrine of Viravain nearby. A rowan sapling clings tenaciously to the ground here.
You see exits leading southeast, west, and northwest.
5225h, 7590m, 6600e, 10p, 32400w elrx<--[5225/5225h]nw
Huddled stands of trees among grass and razor shrubs.
The loud, echoing caw of crows permeates the air, filling the area with a feeling of dread from a
war shrine of Viravain nearby. A blackthorn sapling clings tenaciously to the ground here.
You see exits leading north and southeast.
5225h, 7590m, 6600e, 10p, 32400w elrx<--[5225/5225h]n
```

How awesome is that? Now you'd probably wonder - how did I know that `gmcp.Char.Vitals.hp` corresponds to your current health, and `gmcp.Char.Vitals.maxhp` to your max health? The trick is simple, and you'll see it when you 'unshroud' GMCP - by clicking the 'Debug' button bottom-left:



## Discovering GMCP's data

Suddenly, your screen will be filled with things like this (clicking Debug again will turn it off, by the way):

```
System Message: [REDACTED]
GMCP event <gmcp.Char> display(gmcp) to see the full content
System Message: [REDACTED]
GMCP event <gmcp.Char.Vitals> display(gmcp) to see the full content
The sun reaches the zenith of the firmament, pausing in his quest to
shining golden rays.
5225h, 7590m, 6600e, 10p, 32400w elrx<>- [5225/5225h]
```

The line where it says `GMCP event <gmcp.Char.Vitals>` tells us that the `gmcp.Char.Vitals` table was updated - and because you see it before the prompt, our prompt trigger works and uses the latest information! You might want to look at all that's available in `gmcp.Char.Vitals` besides `hp` and `maxhp` - you can do that by installing the [run Lua code alias](#) and doing `lua gmcp.Char.Vitals`, which will show you all that's available in that table.

```
lua gmcp.Char.Vitals
```

## Recap

So now you know how to see GMCP events as they're coming in (by enabling Debug), how to view what GMCP data you have (by doing `lua gmcp`), and how to retrieve GMCP data (by accessing `gmcp`, like `gmcp.Char.Vitals.hp` will get you your current health). There are two things left to learn: trigger on GMCP events and send GMCP requests to the server (which you can use to ask for more information!)

**Note:** as you might have noticed, Mudlet stores data from incoming GMCP events in the `gmcp` table, with the rest of the key name corresponding to the event name - so if you receive a `Char.Vitals` event from the server, it's data will be stored in `gmcp.Char.Vitals`. Mudlet also raises events whenever you receive GMCP data - which you'll read about later on - and the event name has the same format; `gmcp.Char.Vitals`.

## Sending GMCP

Your MUD might support sending commands via GMCP to it - for examples, take a look at [this sheet](#) for IRE games or [Aardwolf's wiki](#). For example, on IRE games, you can ask the server `Char.Skills.Get` and the server will tell you what skills you have got - this can come in handy for auto-configuring scripts. In Mudlet, the function to send a GMCP request is `sendGMCP()` - so with our example, you'd do:

```
sendGMCP("Char.Skills.Get")
```

in a script and get an appropriate `Char.Skills.Groups` response (as the documentation mentions).

The same `Char.Skills.Get` request can take additional parameters with it, like the documentation mentions:

```
sendGMCP([[Char.Skills.Get { "group": "elemancy", "name": "firelash" }]])
```

(note how I replaced "" with [[]] now - that's because you can't have a " inside a "). You can either use the example above, or you can also use this:

```
sendGMCP("Char.Skills.Get"..yajl.to_string({group = "elemancy", name = "firelash"}))
```

Both have the same effect, but you might find the latter easier to work with as you get better with Lua.

**Note:** There is currently a known issue on IRE games - a response to a GMCP command will not be sent to you until you until the next line, if you have M CCP enabled. So to workaround, you might want to use a blank `send()` command:

```
sendGMCP("Char.Skills.Get"); send("\n")
```

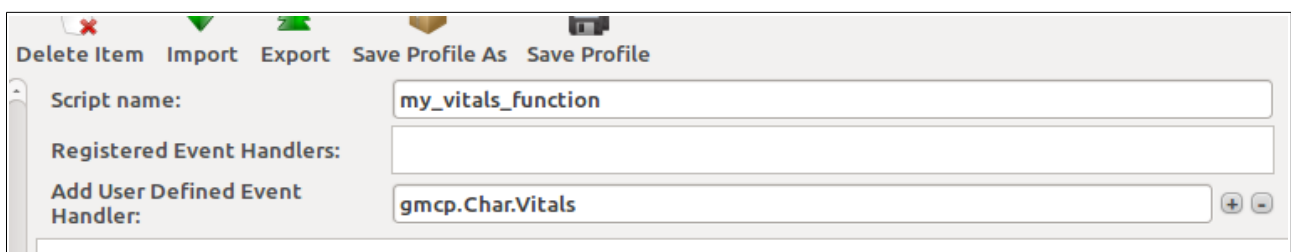
## Triggering on GMCP

Lastly, we'll learn how to trigger on GMCP events - this means doing something right when you got the event, not on some line/prompt from the MUD. Why would you want to do that? There might be times when you receive a GMCP event and an input from the line won't come right away, and you want to act on the event fast - or you might want to do it for neater system organization.

Remember how you unshrouded GMCP by enabling Debug? As an example, we'll make an event trigger on the `gmcp.Char.Vitals` event. Go to Scripts, *Add Item*, and give the script a name, in this instance, 'my\_vitals\_function', and paste:

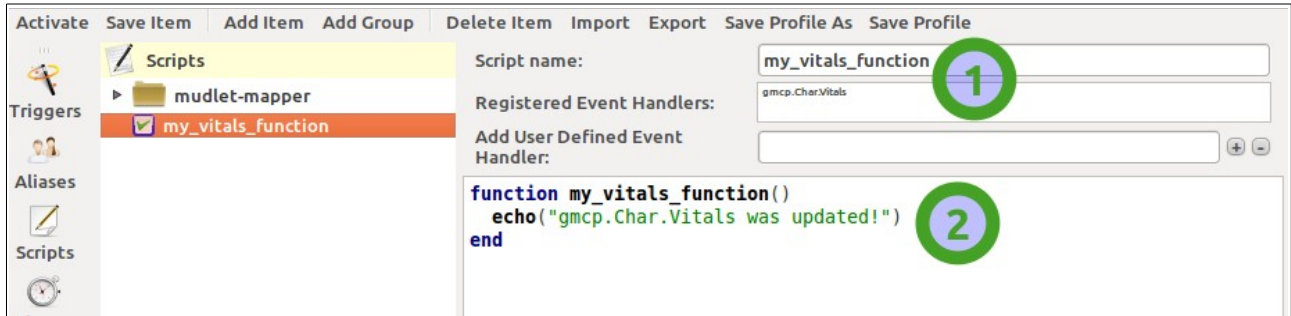
```
gmcp.Char.Vitals
```

into the *Add User Defined Event Handler* field:



The screenshot shows the Mudlet Scripts editor interface. At the top, there are menu options: Delete Item, Import, Export, Save Profile As, and Save Profile. Below the menu, there are three input fields: 'Script name:' with the value 'my\_vitals\_function', 'Registered Event Handlers:' which is empty, and 'Add User Defined Event Handler:' with the value 'gmcp.Char.Vitals'. There are plus and minus buttons next to the last field.

And press enter to add it to the list. This will have Mudlet run whatever function we tell it to when that event comes along - and we'll do that by using the function name as the scripts name, plus defining it in the script



The screenshot shows the Mudlet Scripts editor with the 'Scripts' panel on the left. The 'Scripts' panel shows a folder 'mudlet-mapper' containing a script 'my\_vitals\_function'. The main editor area shows the configuration for the selected script. The 'Script name:' field is 'my\_vitals\_function' (marked with a green circle 1). The 'Registered Event Handlers:' field contains 'gmcp.Char.Vitals'. The 'Add User Defined Event Handler:' field is empty. Below these fields, the script code is displayed: 

```
function my_vitals_function()
  echo("gmcp.Char.Vitals was updated!")
end
```

 (The 'end' line is marked with a green circle 2).

That's it! Now you can add code into your function to do whatever you'd like when you receive a GMCP event.

**Tip:** Mudlet not only raises an event for the exact GMCP event that you receive, but it also raises events for parts of the name, allowing you to create a handler that deals with more than one event, if you find it convenient. So when you receive a `Char.Skills.Groups` event, for example, events `gmcp.Char`, `gmcp.Char.Skills`, and `gmcp.Char.Skills.Groups` events will be raised, in that order.

## Bonus

We can apply our learned GMCP knowledge to create a simple health/mana tracker (this example would be applicable easier to more IRE games than a health sipper). It'll show us our differences in health/mana on each prompt! It's very simple for a purpose – so you can easily see how to improve on it and do it.

Start a new prompt trigger, that has the same setup as the sample one we did previously -

```
return isPrompt()
```

As a *Lua function* pattern type. Next, add this as the code:

```
-- create variables that'll track our previous health, so we can use them to compare
oldhealth = oldhealth or 0
oldmana = oldmana or 0

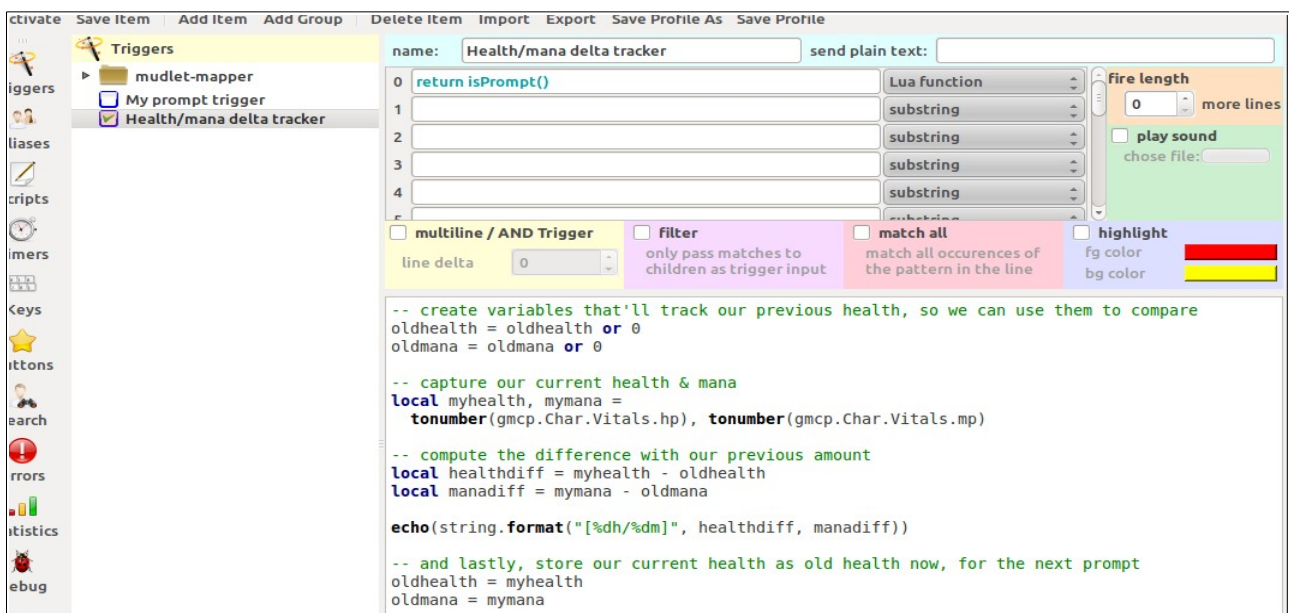
-- capture our current health & mana
local myhealth, mymana =
  tonumber(gmcp.Char.Vitals.hp), tonumber(gmcp.Char.Vitals.mp)

-- compute the difference with our previous amount
local healthdiff = myhealth - oldhealth
local manadiff = mymana - oldmana

echo(string.format("[%dh/%dm]", healthdiff, manadiff))

-- and lastly, store our current health as old health now, for the next prompt
oldhealth = myhealth
oldmana = mymana
```

So it all looks like this:



The screenshot shows the GMCP configuration window. The trigger is named "Health/mana delta tracker" and is set to "send plain text". The trigger pattern is "return isPrompt()". The trigger type is "Lua function". The trigger code is:

```
-- create variables that'll track our previous health, so we can use them to compare
oldhealth = oldhealth or 0
oldmana = oldmana or 0

-- capture our current health & mana
local myhealth, mymana =
  tonumber(gmcp.Char.Vitals.hp), tonumber(gmcp.Char.Vitals.mp)

-- compute the difference with our previous amount
local healthdiff = myhealth - oldhealth
local manadiff = mymana - oldmana

echo(string.format("[%dh/%dm]", healthdiff, manadiff))

-- and lastly, store our current health as old health now, for the next prompt
oldhealth = myhealth
oldmana = mymana
```

That's it. Now you'll see it working below:

```
You have recovered balance on all limbs.  
4900h, 3490m, 23400e, 15900w exd-[0h/0m]  
The silver light grows more intense, and with a gesture of utter submission to the Dragon within,  
you throw your arms wide and your head back as you scream, "Aaassshhaaaaaaxxeeeeiiiiiii!"  
4900h, 3490m, 23400e, 15900w exd-[0h/0m]  
As a distant growl is heard, the silver light coalesces about you...changing you...deforming your  
being...becoming...Greater.  
4900h, 3490m, 23400e, 15900w exd-[0h/0m]  
With an ear-splitting roar, you rear back your draconic head and scream out your triumph.  
You send your falcon winging away to a safe haven.  
6700h, 6240m, 32400e, 29400w exd-[1800h/2750m]lrs  
Your draconic form melts away, leaving you suddenly weaker and more vulnerable.  
Your race is now that of Horkval.  
4900h, 3440m, 23400e, 15900w d-[-1800h/-2800m]  
You have recovered equilibrium.  
4900h, 3440m, 23400e, 15900w ed-[0h/0m]  
You have recovered balance on all limbs.
```

## That's it!

We've covered how to retrieve GMCP data, trigger on GMCP events, and ask for more GMCP information, should covers all of the basics. One last parting tip - not all GMCP modules are activated by default; if you'd like to enable one (like you'd need to do for `Ire.Rift`, for example), check out the [Mudlet wiki](#).

**Tip: You can also utilize GMCP for things like antitheft! Take a look at some examples posted on [Achaean Forums](#).**

Have comments, ran into problems, or want to post a tip? Write below!