


Titan WebAPI

# WebAPI over UDP

By design, the Web API works with HTTP requests - and HTTP is bound to TCP (this is about network

protocols - if you don't know what these acronyms mean then this topic isn't interesting for you ). You may come across situations where you need to control Titan from UDP requests, e.g. when working with particular main control suites.

While it is not possible to simply fire the same request as UDP string onto Titan, it is very well possible to run a little program like socat, which - when started with proper parameters - forwards UDP strings as TCP requests. The only other thing is that you need to send a complete HTTP request string.

## 1. Get Socat

E.g. download it from

[https://sourceforge.net/projects/unix-utils/files/socat/1.7.3.2/socat-1.7.3.2-1-x86\\_64.zip/download](https://sourceforge.net/projects/unix-utils/files/socat/1.7.3.2/socat-1.7.3.2-1-x86_64.zip/download)

## 2. Start Socat

Run Socat on the same computer like Titan, with proper parameters. In my example it worked when called like this in a console window:

```
socat udp4-listen:9090,reuseaddr TCP:192.168.178.59:4430
```

The parameters I used are

- `udp4-listen:9090` - this tells socat to listen to UDP messages, on port 9090, in IPv4 protocol
- `reuseaddr` - allows other sockets to bind to the same address (I found this helps avoiding some issues)
- `TCP:192.168.178.59:4430` - this tells where and how socat forwards the messages to, in out case, as TCP requests, to the IP of the Titan PC (the same computer socat is running on), on port 4430 (the port Titan listens on)

When everything is setup properly then socat just sits there and does its job, without any visual feedback. However when debugging or looking for a particular issue it might help to start socat with some debug output, using the option `-v`

```
socat -v udp4-listen:9090,reuseaddr TCP:192.168.178.59:4430
```

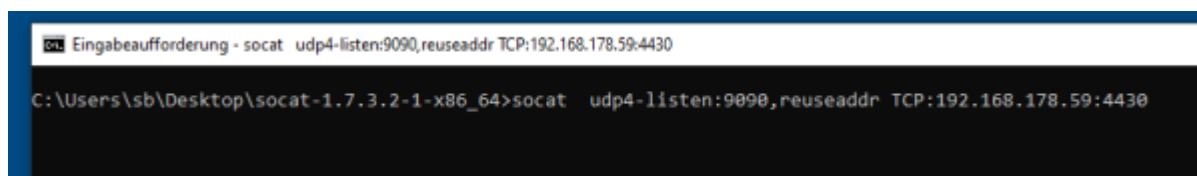
This outputs the transferred data to you screen, and may help in finding possible faults.

You can also start socat with the hostname instead of the IP address, like this:

```
socat udp4-listen:9090,reuseaddr TCP:localhost:4430
```

In order to terminate socat simply use

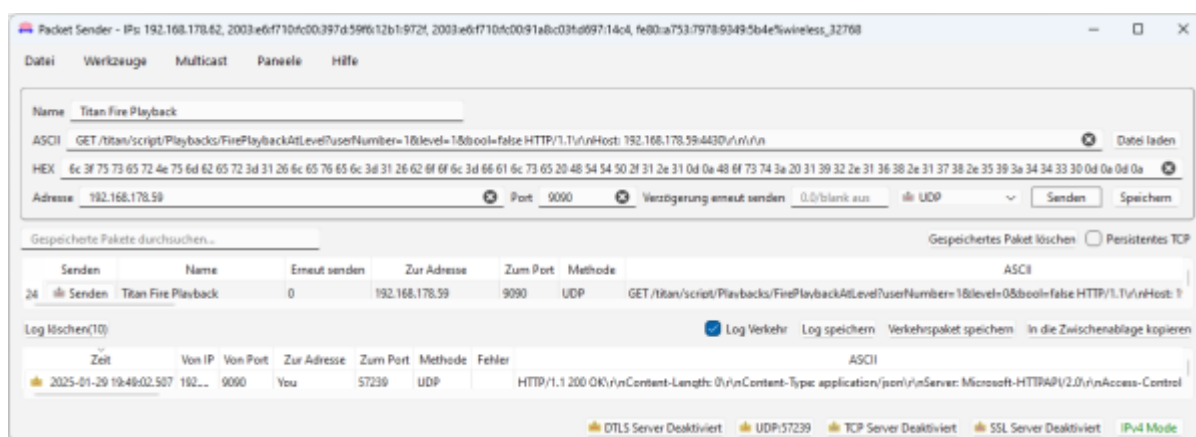
```
Ctrl-C
```



### 3. Send full HTTP request strings over UDP

Now, as socat expects UDP messages and forwards them as TCP, and Titan expects full HTTP requests, we need to send full request strings. Otherwise there will be no result at all, or socat crashes. In my tests the a.m. option -v did help as it shows the reply from the Titan engine (here: Status 400 - Bad Request...).

In order to send some UDP messages, either use your control suite, or - for tests - use a tool like PacketSender (<https://packetsender.com/>).



The parameters to send as UDP are

- address: the IP address of the Titan computer (in my case 192.168.178.59)
- port: 9090 (the port which we told socat to listen to)
- method: UDP
- ASCII string: this needs to be the full HTTP request string, starting with GET, containing the API request itself, the protocol, host/port, and must end with two linefeeds:

```
GET
```

```
/titan/script/Playbacks/FirePlaybackAtLevel?userNumber=1&level=1&bool=false  
HTTP/1.1\r\nHost: 192.168.178.59:4430\r\n\r\n
```

**Only for readability** here is the same string wrapped into lines (but you need to send it as one string):

```
GET
```

```
/titan/script/Playbacks/FirePlaybackAtLevel?userNumber=1&level=1&bool=false  
HTTP/1.1\r\n  
Host: 192.168.178.59:4430\r\n
```

\r\n

1. The first line starts with the HTML method (GET), and contains the API request (/titan/script/Playbacks/FirePlaybackAtLevel?userNumber=1&level=1&bool=false) and the protocol (HTTP/1.1)
2. The second line defines the Host (IP address and port) - this is required to make it a valid HTTP1.1 request
3. The third line simply contains a linefeed (strictly the second linefeed after the one at the end of line 2), in order to mark the end of the request.

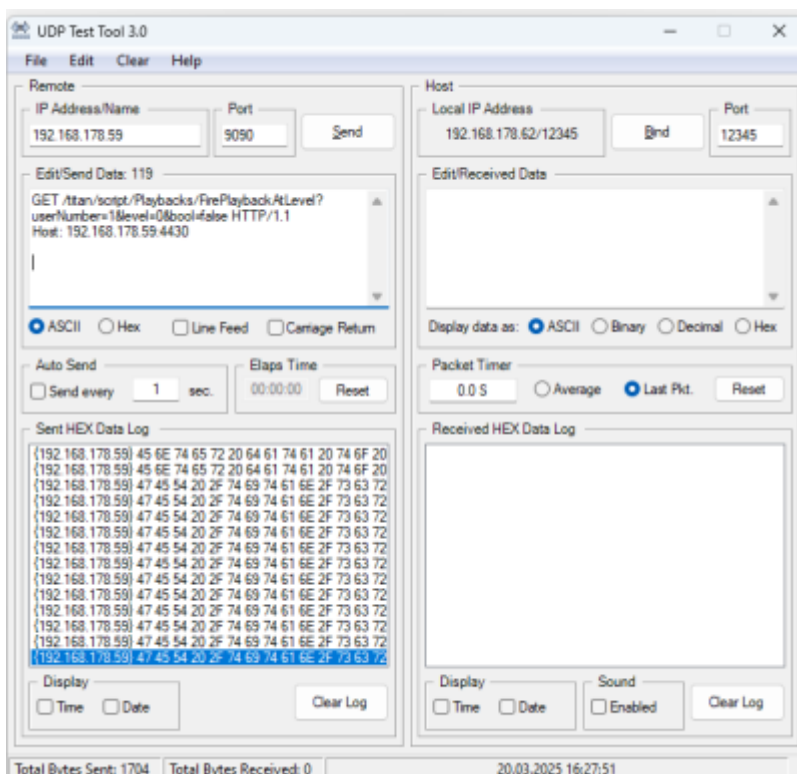
**Note that different UDP programs seem to handle line endings differently.**

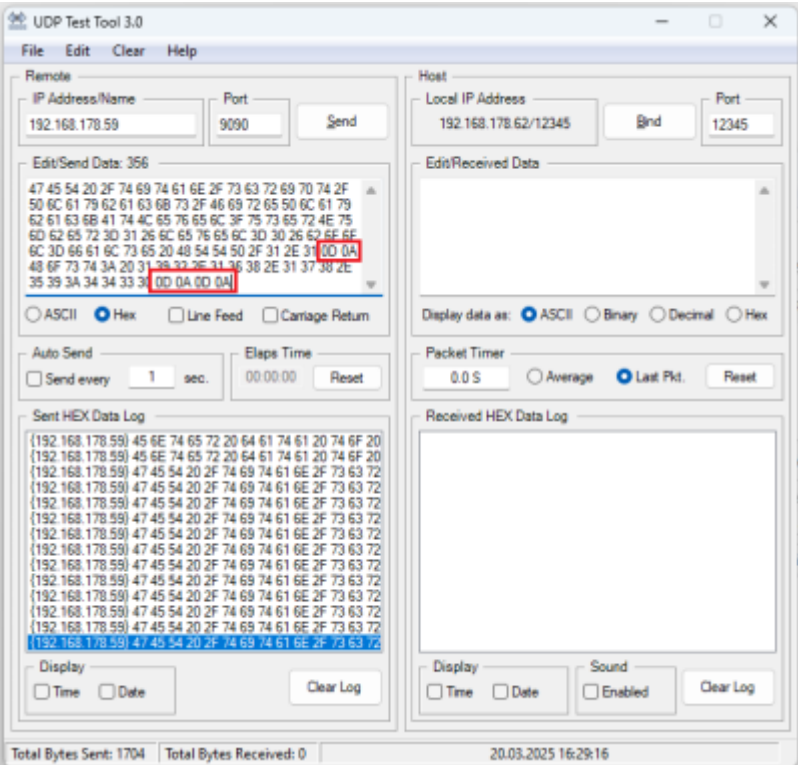
- Microsoft's [UDP-Sender/Receiver](#) seems to be unable to send linefeeds in the transmitted message; if you write them escaped (\r\n) then it sends the literal characters, and a normal linefeed (pressing the ENTER button) isn't reflected in the message at all
- the [UDP Test Tool](#) also sends escaped linefeeds as literal characters, causing Titan to reject this as '400 Bad Request'. However you can press ENTER instead of writing \r\n - and this gets correctly transmitted as linefeed.

Here is a screenshot of the same string entered with pressing ENTER instead of writing the linefeeds:

```
GET /titan/script/Playbacks/FirePlaybackAtLevel?userNumber=1&level=1&bool=false HTTP/1.1
Host: 192.168.178.59:4430
```

Here UDP Test Tool is shown with linefeeds in the ASCII text (hardly visible), and their representation (note the highlighted 0D 0A sequences in the HEX view):





Maybe this does not work for all Web API requests - but it is a starting point to make it available over UDP.

From:  
<https://avosupport.de/wiki/> - **AVOSUPPORT**

Permanent link:  
<https://avosupport.de/wiki/webapi/implementations/udp>

Last update: **2025/03/20 16:08**