

UDP Extension

Level: Advanced

WARNING: Multitplayer games are hard to make. You need to account for data loss / latency / (bad) connection loss / security issues.

Sending and receiving data using this extension is pretty easy, but making a solid game with it is difficult.

Compatible : Stencyl 3.x

Platforms:

- iOS
- Android
- Macintosh OSX
- Windows
- (Not tested : Linux)

Flash : on hold

Important: Firewalls can block UDP connections either on client side or on your side!

UDP = User Datagram Protocol

Implementation of the protocol can vary between platforms but you need to be aware of:

- Packets being send CAN be out of order
- Packets size is small (512 bytes)
- Communication is not guaranteed

If you want data to be send across in order and you rely on it to be received use TCP communication. Don't try to make the UDP protocol safe for ordering and guaranteed communication since that is what the TCP protocol is made for.

Using UDP you get the fastest method of communication on the internet.

Usually this boils down to sending positional data to the server and receive positional data from the server.

See:

https://en.wikipedia.org/wiki/User_Datagram_Protocol

Requirements:

- PHP server (Windows/Linux running PHP > 5.3.0)
- Portforwarding mechanism

- Knowledge on how to run and maintain a server on the network

Basic Installation:

Put the `udp_server.php` file on your dedicated server where you can and should run the socket-server:

```
$ php -f udp_server.php
```

This process needs to be running all the time you want to run/test your game.

Configure PHP

- Port 12000 is default port change that in code when you need/want another
- Make sure that you portforward your router so that UDP data send to port 12000 are being routed to the computer you are running the PHP script on.

Extension documentation:



Server: yourserver.com DO NOT put http or any other protocol in front!

Port: 12000 Change this in the PHP file when you use another port

When you serve more games you can distinguish between them using the AppID.

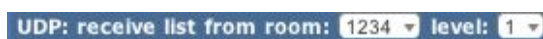
Inside a game it could be beneficial if you limit the area where you communicate between players. The Room can be used to limit the data being send. The Level indicator with send and receive will further limit the data being communicated.

UserID should be unique across your players so that data is send from the correct one. You use the Echo On when the data you send should be send back to you. If you want animations/reactions from your game to match exactly what it does for other players this can be helpful.



The data being send needs to be smaller than 512. Some implementations of the UDP protocol have a limit of 512 therefore it is a limit in the extension. You could experiment with it yourself by adjusting the extension code.

You need to make your own data handling based on the data being communicated!



Actually the data received is stored in memory while it is received but this block is a method of getting the data that has been received.

UDP: receive message on room: 0 level: 0

Data is received in this form: %<userid>=data|data|data%<userid>=data%

Where the “%” is the separator between each user-data information

You need to make your own data-handler to interpret what is received.

Link to the extension:

<https://dl.dropboxusercontent.com/u/107982821/stencyl/UDP/UDP.zip>

Link to the PHP:

https://dl.dropboxusercontent.com/u/107982821/stencyl/UDP/udp_server.php

Testing your UDP server connection availability and speed:

<https://pentest-tools.com/discovery-probing/udp-port-scanner-online-nmap>