## **Protocoll**

#### **Important**

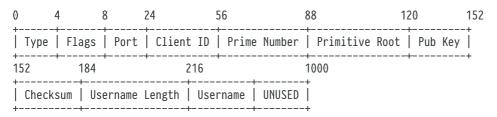
- UDP MTU
- Don't create packages with a size of (bit-size mod 8) != 0. It makes it hard on the receiver side to interpret those!

## General field descriptions

```
Type [4 Bit]:
0000 => Client-Hello-Handshake
0001 => Server-Hello-Handshake
0010 => File-Creation
0011 => File-Transfer
0100 => File-Status
0101 => ACK
0110 => Ping
0111 => Transfer-Ended
1000 => Auth-Request
1001 => Auth-Result
Client ID [32 Bit]:
An unique client id generated by the client on first contact.
E.q. A random int
Checksum [32 Bit]:
CRC32 Algorithm Wiki Link
Sequence Number [32 bit]:
Like TCP.
FID Length [64 Bit]:
The length of the FID field in bytes.
FID [Defined in the FID Length filed]: The relative path to the file. Includes the file name e.q. folder/file.txt.
FID Part Number [32 Bit]:
The file part number.
Pub Key [32 Bit]:
The client public key for the Diffie Hellman encryption.
```

# Client-Hello-Handshake

The initial connection message that gets send by the client.



```
Flags [4 Bit]:

0000
||||
|||+-> Connect requested
||+--> *UNUSED*
+---> *UNUSED*
```

Port [16 Bit]:

The port on which the client listens to server messages.

Prime Number [32 Bit]:

The client prime number for the Diffie Hellman encryption.

Primitive Root [32 Bit]:

The client primitive root for the Diffie Hellman encryption.

Username Length [32 Bit]:

Describes how long the the following Username is in byte.

Username [X Byte]:

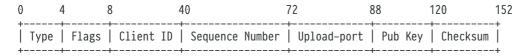
Defined via the Username Length.

UNUSED [816-X Bit]:

To "prevent" DoS attacks. Ensures the package is a least 1000 Bit long.

#### Server-Hello-Handshake

Once the server received a Client-Hello-Handshake message he should reply with this message.



Upload-port [16 Bit]:

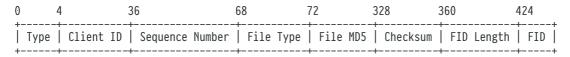
The Port where the client should send all following messages to.

Flags [4 Bit]:

```
0000
||||
|||+→ Client accepted
||+-→ Too many clients - connection revoked
|+--→ Client ID taken - connection revoked
+---→ Invalid username - connection revoked
```

#### File-Creation

Marks the start of a file transfer. Tells the server to create the given file with the given path.



File Type [4 Bit]:

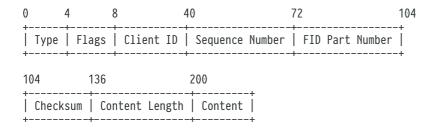


File MD5 hash [256 Bit]:

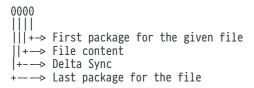
The file MD5 hash to check if the file was transmitted correctly. Unused for folders. Wiki Link

#### File-Transfer

The actual file transfer message containing the file content.



Flags [4 Bit]:

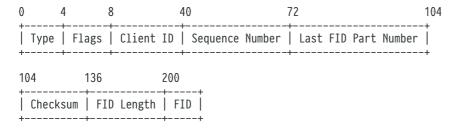


Content Length [max 900 Bit]: The length of the Content field.

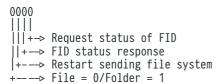
Content [defined in "Content Length" in Bit]: The actual file content.

#### File-Status

Used for requesting and responding the current file status bevor a file gets transfered.



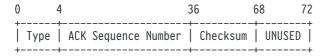
Flags [4 Bit]:



Last FID Part Number [32 Bit]: The last received FID Part Number. Ignored if Request status of FID is set.

#### **ACK**

For acknowledging Ping, File-Creation and File-Transfer messages.

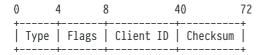


ACK Sequence Number [32 Bit]:

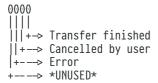
The acknowledged Sequence Number or Ping Sequence Number.

#### Transfer-Ended

Gets send by the client once he wants to end the transfer/close the connection.

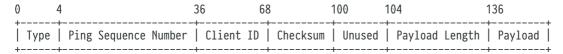


Flags [4 Bit]:



#### Ping

This message is used for ensuring the opponent is still there. The opponent should acknowledge each received Ping message with an Server-ACK. Should get send by each side if there was no message exchange for more than 2 seconds. It also can be used for package loss and throughput tests with a modified Payload Length.



Ping Sequence Number [32 Bit]

An unique number for identifying each ping.

Payload Length [32 Bit]:

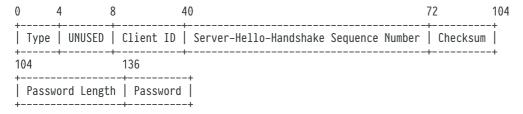
Describes how long the the following payload is in byte.

Payload [X Byte]:

Defined via the Payload Length.

#### **Auth-Request**

Send by the client to authentificate at the server.



Server-Hello-Handshake Sequence Number [32 Bit]:

The sequence number of the Server-Hello-Handshake message.

Password Length [32 Bit]:

Describes how long the the following password is in byte.

Password [X Byte]:

Defined via the Password Length.

#### **Auth-Result**

Send by the server with the authentification result.



# Flags [4 Bit]: 0000 |||| |||+-> Authentification successfull ||+--> \*UNUSED\* |+---> \*UNUSED\* +---> \*UNUSED\*

Auth-Request Sequence Number [32 Bit]: The Sequence number of the received `Auth-Request message.

# Process example

	Server Client-Hello-Handshake	
		The clients starts the connection on the default port and tells the server the port on which he listens for answers. It also contains ```ClientStartConnection``` key exchange data.
	Server-Hello-Handshake	The server responds with an upload port and the
	Auth-Request	```onServerReceive``` key exchange data.
	> Auth-Result	If the client got accepted he sends his password via the now encrypted connection.
	File-Status	The server answers with the result of the authentification.  Now the connection is established.
		The client requests the file status.
	Auth-Result	The server responds with the current file status.
	File-Creation	
	> Server-ACK	The client sends this message if the server does not has the file yet.
		The server acks the ```File-Creation```.
	File-Transfer >	The client starts sending the file in chunks.
	Server-ACK	The convence and on ACV massage for each massage
	File Towns Con	The server sends an ACK message for each message it received from the client
	File-Transfer →	The client sets the ```Last package for the file``` flag
	Server-ACK	to inform the server, it is the last file part.
	Transfer-Ended >	The client tells the server that he liks to close the connection.