# Wave Messaging Protocol

Use Protobuf 3 to encode message.

## Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Value** | **Direction of flow** | **Description** |
| Reserved | 0 |  |  |
| CONNECT | 1 | Client to Server | Client request to connect to Server |
| CONNACK | 2 | Server to Client | Connect acknowledgment |
| PUBLISH | 3 | Client to Server or Server to Client | Publish message |
| PUBACK | 4 | Client to Server or Server to Client | Publish acknowledgment |
| SUBSCRIBE | 5 | Client to Server | Client subscribe request |
| SUBACK | 6 | Server to Client | Subscribe acknowledgment |
| UNSUBSCRIBE | 7 | Client to Server | Unsubscribe request |
| UNSUBACK | 8 | Server to Client | Unsubscribe acknowledgment |
| PINGREQ | 9 | Client to Server | PING request |
| PINGRESP | 10 | Server to Client | PING response |
| DISCONNECT | 11 | Client to Server |  |

First byte lowest 4 bits

## Version

First byte highest 4 bits

## WMP Control Packets

### CONNECT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Type | Range | Required | Description |
| clientId | int64 | 0 ~ 2^63-1 | True | Client Unique Id |
| token | string | 0 ~ 100(byte) | True | Token generated by server after client login |

**Protobuf:**

message WmpConnectMessageBody {

string clientId = 1;

string token = 2;

}

### CONNACK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Type | Range | Required | Description |
| clientId | int64 | 0 ~ 2^63-1 | True | Client Unique Id |
| returnCode | enum | 0 ~ 100(byte) | False | WmpConnectReturnCode |

**Protobuf:**

message WmpConnAckMessageBody {

string clientId = 1;

WmpConnectReturnCode returnCode = 3;

}

enum WmpConnectReturnCode {

ACCEPTED = 0;

REFUSED\_UNACCEPTABLE\_PROTOCOL\_VERSION = 1;

REFUSED\_IDENTIFIER\_REJECTED = 2;

/\*\* not login or error token \*/

REFUSED\_NOT\_AUTHORIZED = 3;

/\*\* server is busy or not able to provide service\*/

REFUSED\_SERVER\_UNAVAILABLE = 4;

}

### PUBLISH

message WmpPublishMessageBody {

string content = 1;

/\*\* clientId or topic name, if many elements, separated by coma(,) \*/

string target = 2;

/\*\* one or many --> a clientId or a topic contains many clientIds \*/

TargeType targetType = 3;

/\*\* message from where to where\*/

Direction direction = 4;

}

/\*\* a chat group or a client tag or other self defined group \*/

enum TargeType {

/\*\* one client \*/

CLIENT\_ID = 0;

TOPIC\_GENERAL = 1;

}

enum Direction {

/\*\* message going to client sdk \*/

TO\_CLIENT\_SDK = 0;

/\*\* message going to handler \*/

TO\_SERVER\_HANDLER = 1;

}

### SUBSCRIBE

### SUBACK

### UNSUBSCRIBE

### UNSUBACK

### PINGREQ

### PINGRESP

### DISCONNECT

message WmpDisConnectMessageBody {

string clientId = 1;

}