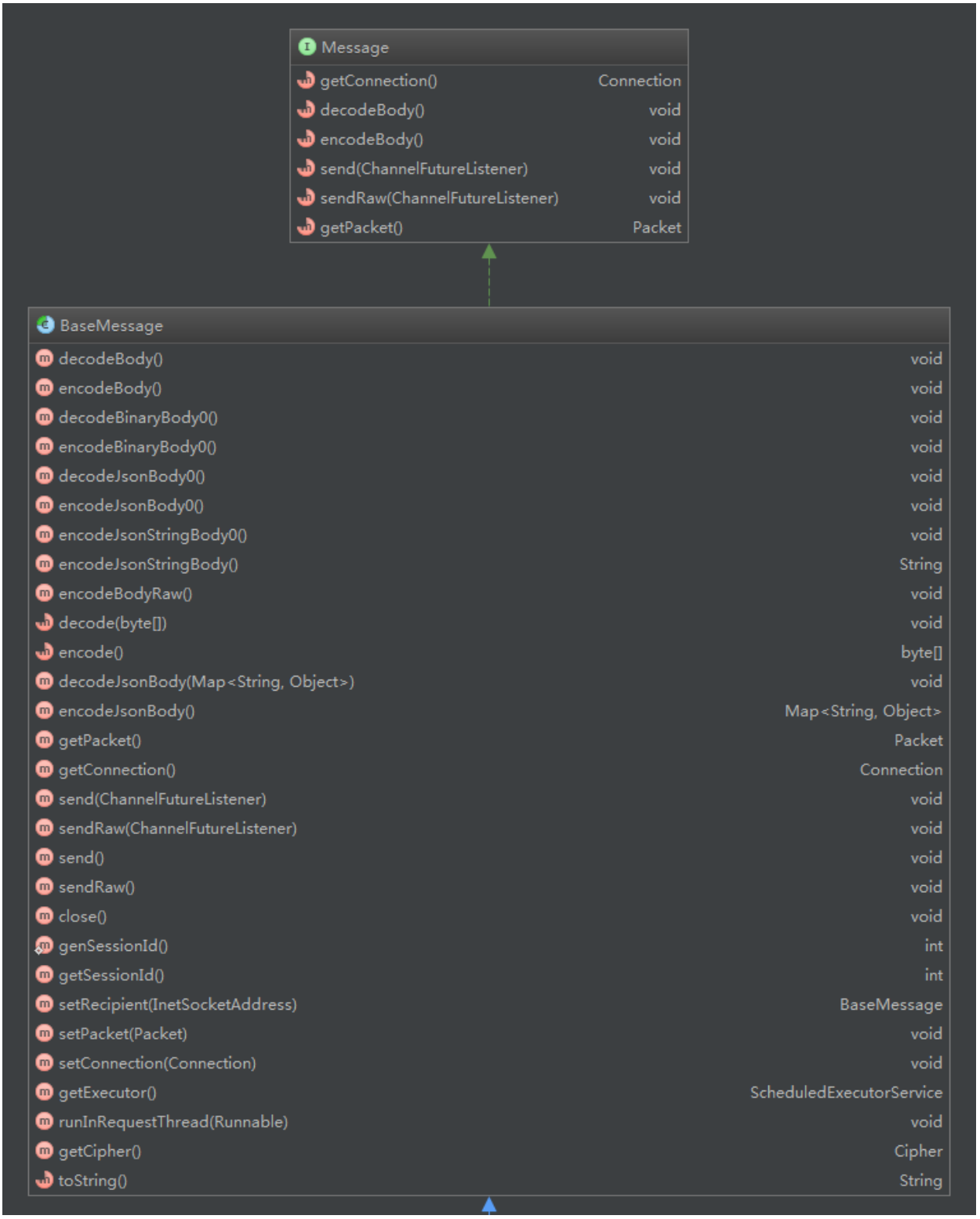
1、接收网关服务的推送消息

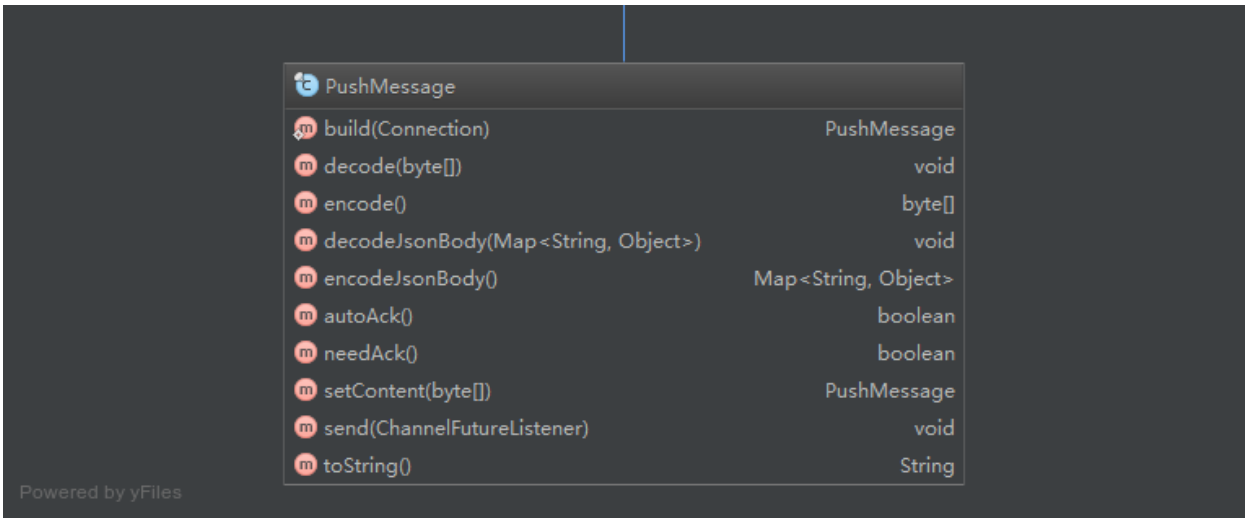
2、推送消息到服务端(暂时没找到相关实现代码)

接收网关服务的推送消息

```
1  //ConnClientChannelHandler.java
2
3  @Override
4  public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exc
eption {
5   connection.updateLastReadTime();
6   if (msg instanceof Packet) {
7   Packet packet = (Packet) msg;
8   Command command = Command.toCMD(packet.cmd);
9   if (command == Command.HANDSHAKE) {
10   ...
11   } else if (command == Command.FAST_CONNECT) {
12   ...
13   } else if (command == Command.KICK) {
14   ...
15   } else if (command == Command.ERROR) {
16   ...
17   } else if (command == Command.PUSH) {
18   int receivePushNum = STATISTICS.receivePushNum.incrementAndGet();
19   PushMessage message = new PushMessage(packet, connection);
20   message.decodeBody();
21   //TODO 根据推送内容content，做业务处理
22   LOGGER.info("receive push message, content={}, receivePushNum={}"
23   , new String(message.content, Constants.UTF_8), receivePushNum);
24   //如果需要ACK，则发送ACK消息
25   if (message.needAck()) {
26   AckMessage.from(message).sendRaw();
27   LOGGER.info("send ack success for sessionId={}",
message.getSessionId());
28   }
29   } else if (command == Command.HEARTBEAT) {
30   LOGGER.info("receive heartbeat pong...");
31   } else if (command == Command.OK) {
32   ...
33   } else if (command == Command.HTTP_PROXY) {
```

```
34      ...
35      }
36      }
37      LOGGER.debug("receive package={}, chanel={}", msg, ctx.channel());
38  }
39
40
41
```






```java
public final class PushMessage extends BaseMessage {

 public byte[] content;

 public PushMessage(Packet packet, Connection connection) {
 super(packet, connection);
 }

 public static PushMessage build(Connection connection) {
  if (connection.getSessionContext().isSecurity()) {
  return new PushMessage(new Packet(PUSH, genSessionId()), connection);
  } else {
  return new PushMessage(new JsonPacket(PUSH, genSessionId()),
connection);
  }
  }

  @Override
  public void decode(byte[] body) {
  content = body;
  }

  @Override
  public byte[] encode() {
  return content;
  }

  @Override
  public void decodeJsonBody(Map<String, Object> body) {
```

```
29  String content = (String) body.get("content");
30  if (content != null) {
31  this.content = content.getBytes(Constants.UTF_8);
32  }
33  }
34
35  @Override
36  public Map<String, Object> encodeJsonBody() {
37  if (content != null) {
38  return Collections.singletonMap("content", new String(content, Constant
s.UTF_8));
39  }
40  return null;
41  }
42
43  public boolean autoAck() {
44  return packet.hasFlag(Packet.FLAG_AUTO_ACK);
45  }
46
47  public boolean needAck() {
48  return packet.hasFlag(Packet.FLAG_BIZ_ACK) || packet.hasFlag(Packet.FLA
G_AUTO_ACK);
49  }
50
51  public PushMessage setContent(byte[] content) {
52  this.content = content;
53  return this;
54  }
55
56
57
58  @Override
59  public void send(ChannelFutureListener listener) {
60  super.send(listener);
61  this.content = null;//释放内存
62  }
63
64  @Override
65  public String toString() {
66  return "PushMessage{" +
67  "content='" + content.length + '\'' +
```

```
68   ", packet=" + packet +
69   '}';
70   }
71 }
```