

- 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 Exception {
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={}",
28                     message.getSessionId());
29             }
30             } else if (command == Command.HEARTBEAT) {
31                 LOGGER.info("receive heartbeat pong...");
32             } else if (command == Command.OK) {
33                 ...
34             } else if (command == Command.HTTP_PROXY) {
```

```

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

```

Message	
getConnection()	Connection
decodeBody()	void
encodeBody()	void
send(ChannelFutureListener)	void
sendRaw(ChannelFutureListener)	void
getPacket()	Packet

BaseMessage	
decodeBody()	void
encodeBody()	void
decodeBinaryBody0()	void
encodeBinaryBody0()	void
decodeJsonBody0()	void
encodeJsonBody0()	void
encodeJsonStringBody0()	void
encodeBodyRaw()	String
decode(byte[])	void
encode()	byte[]
decodeJsonBody(Map<String, Object>)	void
encodeJsonBody()	Map<String, Object>
getPacket()	Packet
getConnection()	Connection
send(ChannelFutureListener)	void
sendRaw(ChannelFutureListener)	void
send()	void
sendRaw()	void
close()	void
genSessionId()	int
getSessionId()	int
setRecipient(InetSocketAddress)	BaseMessage
setPacket(Packet)	void
setConnection(Connection)	void
getExecutor()	ScheduledExecutorService
runInRequestThread(Runnable)	void
getCipher()	Cipher
toString()	String

PushMessage	
m build(Connection)	PushMessage
m decode(byte[])	void
m encode()	byte[]
m decodeJsonBody(Map<String, Object>)	void
m encodeJsonBody()	Map<String, Object>
m autoAck()	boolean
m needAck()	boolean
m setContent(byte[])	PushMessage
m send(ChannelFutureListener)	void
m toString()	String

Powered by yFiles

```

1 public final class PushMessage extends BaseMessage {
2
3     public byte[] content;
4
5     public PushMessage(Packet packet, Connection connection) {
6         super(packet, connection);
7     }
8
9     public static PushMessage build(Connection connection) {
10         if (connection.getSessionContext().isSecurity()) {
11             return new PushMessage(new Packet(PUSH, genSessionId()), connection);
12         } else {
13             return new PushMessage(new JsonPacket(PUSH, genSessionId()),
14                 connection);
15         }
16
17         @Override
18         public void decode(byte[] body) {
19             content = body;
20         }
21
22         @Override
23         public byte[] encode() {
24             return content;
25         }
26
27         @Override
28         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, Constants.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.FLAG_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 }
```