Message Bus

1 RabbitMQ-server Configuration

SuperMap iMobile for Android&iOS provides the mobile terminal message bus module. This module supports the delivery of messages from a specified server at the mobile terminal, the submit of data from a mobile terminal to a specified server, and the distribution of data from a mobile terminal to a specified number of terminals. SuperMap iMobile message bus modules requires constructing RabbitMQ-server. This chapter will introduce RabbitMQ-server configuration and usage.

1.1 Environment Configuration

1.1.1 Install ERLANG Language Package

First, go to http://www.erlang.org/download.html, download Erlang Windows Binary File and run it. It will take around 5 minutes to install it.

The installation steps are as follows:

- Double click otp_win32_R16801.exe (different versions may have different names), and select next
- It will be installed at the C disk by default. It is suggested you install the program on the non-system disk such as D disk (if you install it on the C disk, there might be some permission problems occur). The installation path cannot contain spaces. After modifying the installation path, select next
- Enter the installation program, click install to finish the installation.

1.1.2 Install RabbitMQ Server Software

Go to the page below:

Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

http://www.rabbitmq.com/releases/rabbitmq-server/v3.1.5/rabbitmq-server-3.1.5.exe, and then run and install.

The installation steps are as follows:

- Double click rabbitmq-server-3.1.1.exe, and select next
- It will be installed on the C disk by default. You can directly install it.
- It will take around 2 minutes to finish the installation

1.1.3 Configure Environment Variables

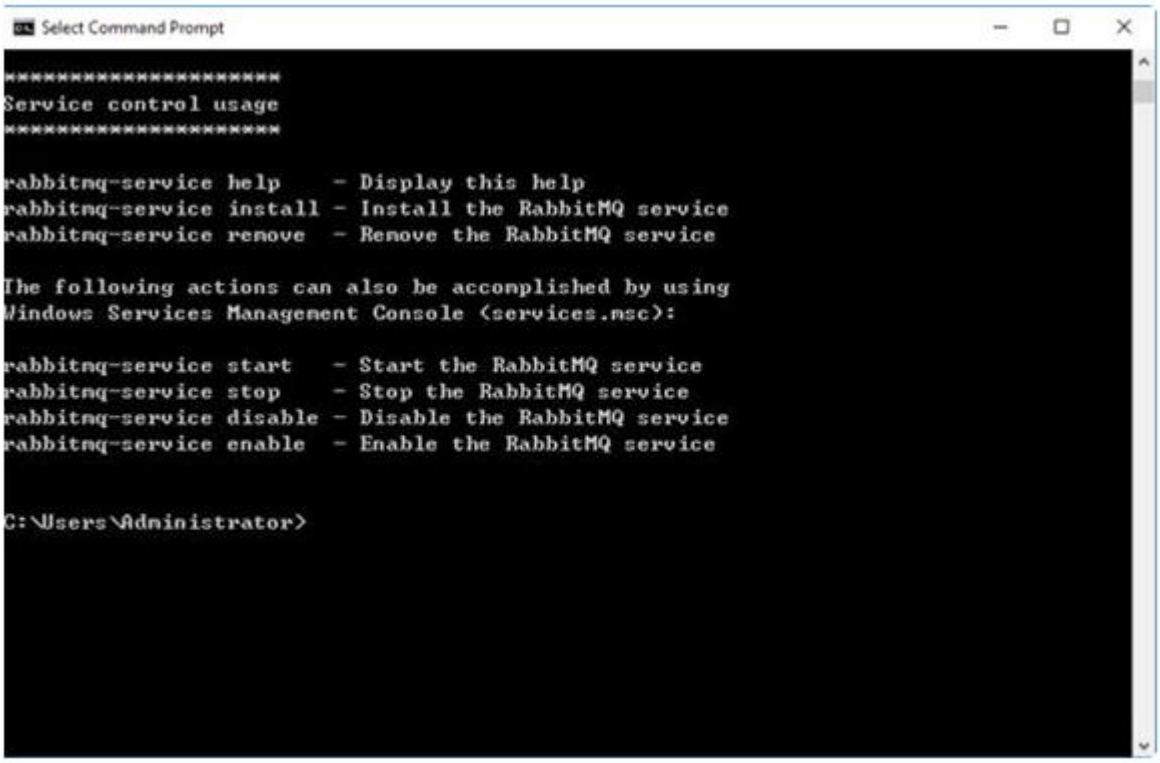
If you need Rabbitmq-sever to run with the windows command line, you still need to configure following environment variables.

Add the RABBITMQ_SERVER environment variable with the value being the installation path of rabbitmq-server. The wizard will install the program to the C disk, and the detailed path is:

C:\Program Files (x86)\RabbitMQ Server\rabbitmq_server-3.5.1

Add %RABBITMQ_SERVER%\sbin to Path, and then enable the windows command line (“cmd”), input rabbitmq-service. If following prompts display, it indicates the environment variables have been configured successfully.

Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com



1.2 RabbitMQ-server Usage

The following commands are based on the command line.

1.2.1 Enable Monitoring Function

rabbitmq-plugins enable rabbitmq_management Enables the monitor management, and then enable the Rabbitmq server. Open the website http://localhost:15672/, with the user name and password being both guest.

1.2.2 Enable Server

rabbitmq-service start

1.2.3 Stop Server

rabbitmq-service stop

Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

1.2.4 Install

rabbitmq-service install

1.2.5 Verification

The way to verify whether the server has been configured successfully is simple. Open the browser, input http://localhost:15672/. If the following page display, it indicates the server has been configured successfully.

The default user is guest, and the password is: guest

SuperMap iMobile 9D for Android Knowledge Base
Message Bus

2 AMQP Protocol Based Message Bus Client Solution

The message bus module offered by SuperMap iMobile for Android&iOS supports AMQP, MQTT, STOMP protocols. This chapter will introduced how to implement message communication via the three protocols.

Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

2.1 AMQP Management Class

AMQPManager of AMQP is responsible for the creation and binding of queues, switches, receivers, and senders.

You can implement the message transceiver function within the queue by binding the queues and switches and setting up unique RoutingKey.

Note: While being used, the receiver needs to receive the message in the child thread, and the received message is blocked.

```
private AMQPReceiver mReceiver_Message = null;

public boolean MessageQueue() {

//Construct AMQPManager

AMQPManager mAMQPManager = new AMQPManager();

//Build connection to server

mAMQPManager.connection(sIP,sPort,sHostName,sUserName,sPassword,sUserId);

//Declare switch

mAMQPManager.declareExchange(sExchange, AMQPExchangeType.TOPIC);

//Declare queue

mAMQPManager.declareQueue(sQueue_Message);

Construct AMQP sender

AMQPSender mAMQPSender = mAMQPManager.newSender();

Construct AMQP receiver

AMQPReceiver mReceiver_Message =
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

```
mAMQPManager.newReceiver(sQueue_Message);

//Bind queue

mAMQPManager.bindQueue(sExchange,sQueue_Message,sRoutingKey_Message);

}
```

2.2AMQP Sender

AMQPSender of AMQP is responsible for sending messages of AMQP.

```
public boolean sendMessage(String geoJson) {

        if(mAMQPSender != null)

         {

            if (geoJson.isEmpty()) {

            return true;

            }

           else {

           boolean bSend = false;

           //Send messages

           bSend = mAMQPSender.sendMessage(sExchange, geoJson, sRoutingKey_TxtMessage);

           }

           System.out.println("send:"+bSend);
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

```
        }
    else
    {
        System.out.println("No connection has been made, please create the connection");
        return false;
      }
  return true;
}
```

2.3 **AMQP Receiver.**

AMQPReceiver of AMQP is responsible for receiving messages of AMQP.

```
private void startReceiveMessage() {
    if (m_Thread_Message == null) {
        m_Thread_Message =  new Thread(new Runnable() {
         @Override
         public void run() {
          Looper.prepare();
            while (mReceiver_Message != null) {
                //Receive messages
                AMQPReturnMessage returnMsg = mReceiver_Message.receiveMessage();
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

```
                if (!returnMsg.getMessage().isEmpty()) {

                    if (returnMsg.getQueue().equals(sUserId)) {

                        continue;// Message sent by itself, ignore it

                    }

                    // Get the message

                  String msg = returnMsg.getMessage();

                }

            }

          }

        });

        m_Thread_Message.start();

    }

}
```

2.4 AMQP Switch Type

AMQP protocol supports 3 types of switches.

- **DIRECT:** Redirect messages to the queue specified in routingKey. It requires that the bindingKey used for queue binding and the routingKey used for message sending are consistent, ensuring only queues matched with key can receive and send messages.
- **FANOUT:** Forward messages to all queues that are bound to the switch. If the receiver and the sender use the same switch, all the clients can send and receive messages
- **TOPIC:** Forward messages to all queues that cares the specified topic in routingkey. Only if the topic the queue cares (bindingkey) can fuzzy match the routingkey of the message, the message can be sent to the queue.

SuperMap iMobile 9D for Android Knowledge Base

Message Bus

Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

3 MQTT Protocol Based Message Bus Client Solution

3.1 MQTT Client

MQTTClient of MQTT realizes message receiving and sending via subscribing topics.

```
private int qos1 = 2;

private List<String> msgList = new ArrayList<String>();

public void MessageQueue () {

threadFlag = true;

//Construct MQTT client

MQTTClient mqttAndroidClient1 = new MQTTClient();

MQTTClient mqttAndroidClient2 = new MQTTClient();

//Establish connection

mqttAndroidClient1.create("182.92.150.115","supermap","supermap123", clientID[0]);

mqttAndroidClient2.create("182.92.150.115","supermap","supermap123", clientID[1]);

//Subscribe to a topic

mqttAndroidClient1.subscribe(topicName, qos1);

//Receive messages

thread = new Thread(new Runnable() {

    @Override
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

```
    public void run() {
      while (threadFlag) {
      //Receive messages
      String expReceive1 = revMessage(mqttAndroidClient1);
      if (!expReceive1.isEmpty()) {
          msgList.add(expReceive1);
      }
    }
 }
});
  thread.start();
  //Send messages
  mqttAndroidClient2.sendMessage(topicName, sMessage);
 }
//Receive messages
private String revMessage(MQTTClient mqttClient) {
    mqReturnMessage = mqttClient.receiveMessage();
    String receiveMsg = mqReturnMessage.getMessage();
    return receiveMsg;
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

```
        }
```

4 STOMP Protocol Based Message Bus Client Solution

4.1 STOMP Management Class

STOMP managemnet class is responsible for initializing libraries, establishing connections, creating a sender, creating a receiver, shutting down libraries.

```
private static STOMPManager stompManager;

private static STOMPSender stompSender;

private static STOMPReceiver stompReceiver;

private String topicName = "testTopic";

public void MessageQueue () {

//Initialize libraries

STOMPManager.initializeLibrary();

//Construct STOMPManager

stompManager = new STOMPManager();

//Establish connection

stompManager.connection(

"failover:(tcp://192.168.18.179:61613?wireFormat=stomp)",

"supermap", "supermap123");

//Create receiver

stompReceiver = stompManager.newReceiver(true, topicName, clientID[0]);
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

```
//Receive messages
 thread = new Thread(new Runnable() {
      @Override
        public void run() {
          // TODO Auto-generated method stub
          while (threadFlag) {
          //Receive messages
          String expReceive = stompReceiver.receive();
          }
        }
 });
 thread.start();
//Create sender
stompSender = stompManager.newSender(true, topicName);
//Send messages
stompSender.sendMessage(sMessage);
}
```


Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com

4.2 STOMP Sender

STOMPSender of STOMP is used to send messages of the STOMP service (for details, please refer to 4.1).

4.3 STOMP Receiver

STOMPReceiver of STOMP is used to receive messages of the STOMP service (for details, please refer to 4.1).

Add: 6/F, Building 107, No. A10, Jiuxianqiao North Road, Chaoyang District, Beijing, 100015, CHINA, 100015
E-mail: request@supermap.com Website: www.supermap.com