**Final Project Demo Notes**

Chat

Introduction

QuickBlox Chat is a quick and reliable chat solution which combines benefits of scalable cloud hosted XMPP chat server, seamless Single Sign-On authorization via Users, incoming IM / chat alerts via Push Notifications and file attachments via Content.

Server

Main server: chat.quickblox.com.

MUC (Multi Users Chat): muc.chat.quickblox.com. Important: users registration, authentication and removal must be done via Users API (and not XMPP).

Login / ID

In order to login to chat you have to create a user. You can do it on QuickBlox dashboard or via API.

Each user gets a JID (Jabber ID) in the following format:

<user\_id>-<app\_id>@chat.quickblox.com

Password

Your user’s password for XMPP connection depends on what type of user authentication via Users you have applied for this particular user:

standard login+password authentication: use same password

Facebook/Twitter/Twitter Digits authentication: use session token as password:

REST API

Chat REST API provides an access to Chat history. We operate with 2 models: Chat Dialog and Chat Message.

Dialog model

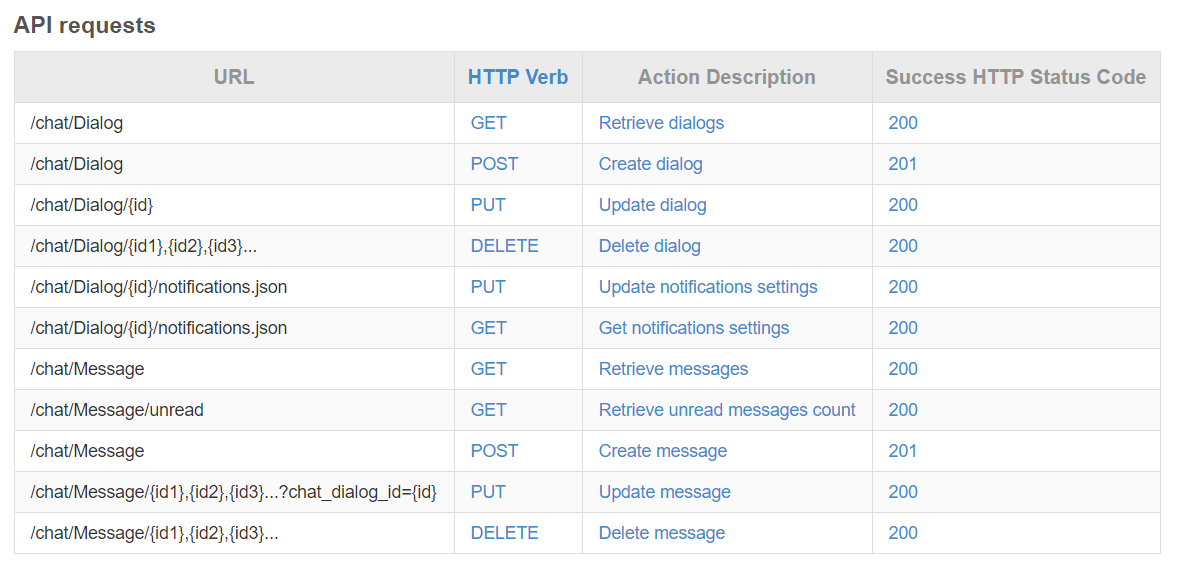
Chat Dialog model describes a dialog entity between users (1-1 chat or group chat). Dialog can have custom parameters (described here)

e.g. Type of dialog. Possible values: 1(PUBLIC\_GROUP), 2(GROUP), 3(PRIVATE), Name of a group chat, Photo of a group chat, Last sent message in this dialog...

Message model

Chat Message model describes a chat message in a dialog

e.g. Message body, Message date sent, Message sender ID, Message recipient ID...



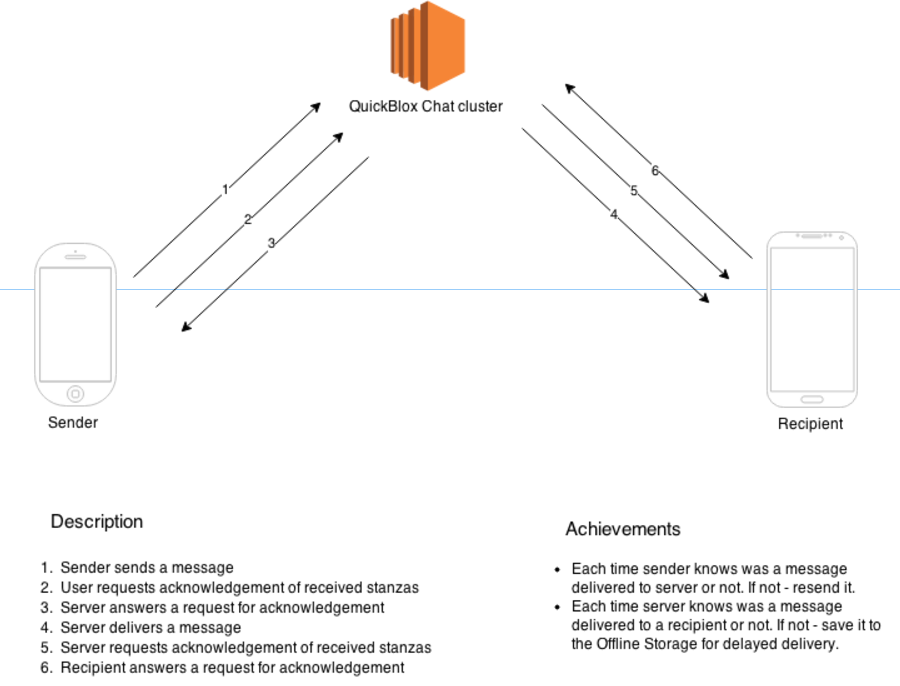
Stream management

Stream management (XEP-0198) defines an XMPP protocol extension for active management of an XML stream between two XMPP entities, including features for stanza acknowledgements and stream resumption.

The basic concept behind stream management is that the initiating entity (either a client or a server) and the receiving entity (a server) can exchange "commands" for active management of the stream. The following stream management features are of particular interest because they are expected to improve network reliability and the end-user experience:

* Stanza Acknowledgements -- the ability to know if a stanza or series of stanzas has been received by one's peer.
* Stream Resumption -- the ability to quickly resume a stream that has been terminated.

The main interesting part here is **Stanza Acknowledgements** because it allows to achieve 100% reliability. To better understand how it works we prepared next picture:



MainActivity.onCreate()

checkRegisration->yes->checkImportFriends()

->no ->loginChat()(到登入頁面)

launchDialogsListFragment()->DialogsListFragment.newInstance()(到主頁面)

DialogsListFragment. onOptionsItemSelected()

點search item-> SearchFragment.newInstance()(到search頁面)

RadioGroupListener.onChckedChange()->local or global

SearchViwPagerAdapter.getSearchListener(0or1)

onQueryTextSubmit ->search(傳入要找的名字)

點chat list item->Private type->startPrivateChatActivity(傳入點擊的dialog)

->Private type->startGroupChatActivity(傳入點擊的dialog)

PrivateDialogActivity.start(傳入點擊的dialog與對話oppoment)

BaseDialogActivity()

messageEditTextChanged()(EditText狀態有變化就call isTypingNow立刻改變狀態)-> sendTypingStatus()(送出typing狀態給server)

BaseDialogActivity extends BaseLoggableActivity implements

EmojiconGridFragment.OnEmojiconClickedListener, EmojiconsFragment.OnEmojiconBackspaceClickedListener,

ChatUIHelperListener, OnImagePickedListener

send message:

onTextChange message\_edittext->如果訊息欄是空的就顯示attach button不是空的就顯示send button(call setVisibility(Gone or Visible))->onClick send button->call sendMessage()用傳入的boolean判斷是private還是group->是private就call sendPrivateMessage 是group就call sendGroupMessage->是private就傳入message和對方的userId 是group就傳入roomId和messege

Private:

取得dialogId->將time和dialogId加入message的property->用userId來getPrivateChat->根據dialogId決定要把message送去哪裡->call sendPacket()(XMPP connenction)->更新dialog最後更新時間

(API: package org.jivesoftware.smack)

Group:

取得dialogId->用roomId來getGroupChat->將time和dialogId加入message的property->根據dialogId決定要把message送去哪裡->call sendPacket()(XMPP connenction)->更新dialog最後更新時間

(API: package org.jivesoftware.smack)

attach file:

onClick attach button->show ImageSourcePickerDialog->進到一個DialogFragment->onCreateDialog裡判斷我們點到哪個item(gallery or camera)->在OnImageSourcePickedListener依據點到的是哪種來決定要call startImagePicker或是startCameraForResult-> 選camera就到getTemporaryCameraFile()去抓photoFile，選gallery就到getCreatedFileFromUrl

smile panel:

onClick smile panel->先判斷目前smile layout是開還是關，開的話就把panel關上 keyboard打開，關的話就把panel打開 keyboard關上->用onEmojiconClicked來判斷按下了哪個貼圖->按下去後把貼圖放到訊息編輯欄

(API: package com.rockerhieu.emojicon)