邮件推送

SMTP 文档

腾讯云

文档目录

SMTP 文档
SMTP 发送邮件指南

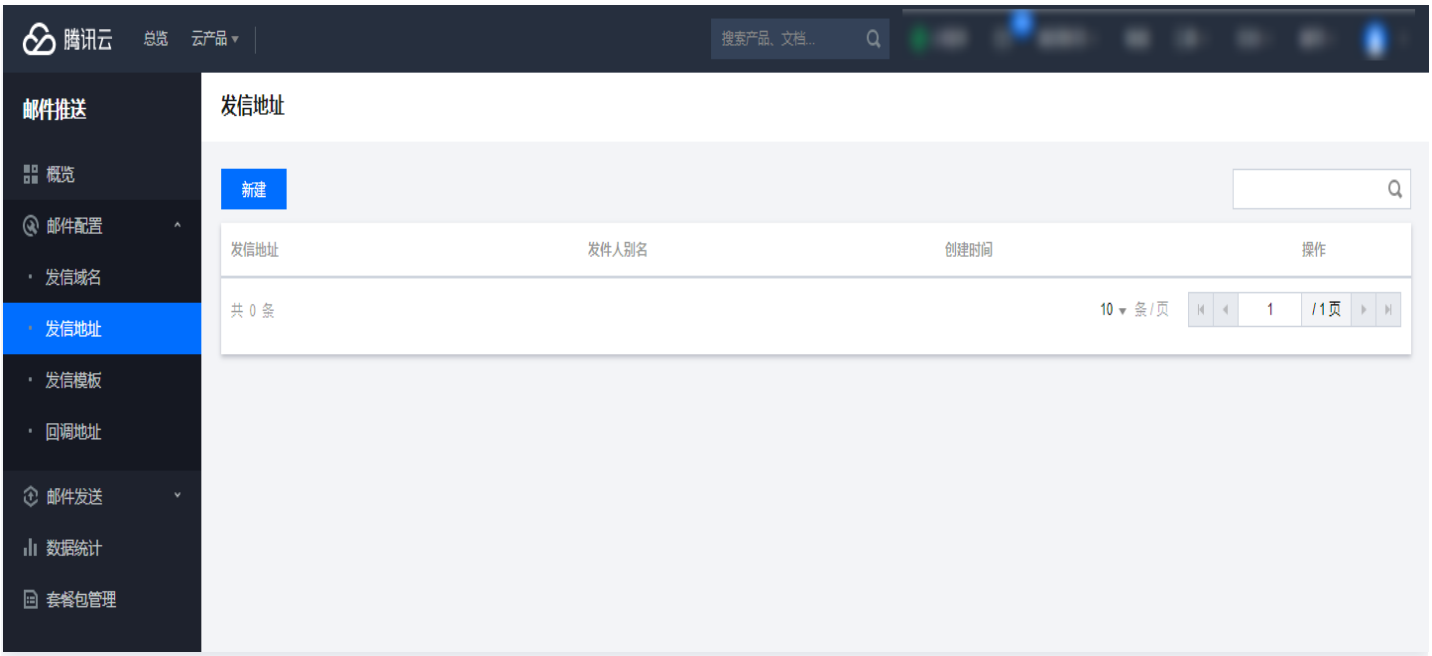
最近更新时间：2022-02-16 10:16:18

开启 SMTP 发信功能

操作步骤

步骤1：进入发信地址

登录 邮件推送控制台 ，单击左侧导航栏中 邮件配置-发信地址，进入发信地址页面。



步骤2：配置 SMTP 密码

1. 在发信地址列表中，找到您要开启 SMTP 发信功能的发信地址，在对应的操作栏中单击**设置 SMTP 密码**。
2. 在弹出对话框中填写 SMTP 密码，然后单击**确定**。

使用 SMTP 接口发信

SMTP 调用方法示例，及具体请求参数，返回参数，错误码说明，请参见 SMTP 调用示例 。

> **说明**
> 支持发送带附件的邮件，具体请参见 SMTP 发送带附件的邮件 。

使用 SMTP 发信频率

目前 SMTP 接口调用频率限制为：同一个 appId 发信频率为20/1s (appId 即腾讯云账号的 appId)。同时同一发信人对同一收信人发信频率限制为10/1h。
我们将收到邮件后，将尽快的将邮件投递出去，但由于各个邮件系统的限流策略，声誉保护策略不同，为了提供您的邮件递送成功率，请尽量以较低的频率发信。

SMTP 服务地址

最近更新时间：2023-05-15 14:49:14

SMTP 服务地址如下：

地址	说明
SMTP 服务地址（中国站香港区），腾讯云香港区客户使用。	smtp.qcloudmail.com
SMTP 服务地址（中国站广州区），腾讯云广州区客户使用。	gz-smtp.qcloudmail.com
SMTP 服务地址（国际站），腾讯云国际站客户使用。	sg-smtp.qcloudmail.com

SMTP 端口号如下：

端口	说明
SMTP 端口号	465（SSL 加密）

说明

基于安全考虑，目前已禁用 25 端口。

Java 调用示例

最近更新时间：2024-04-09 17:52:51

以下代码示例，是 Demo 使用 JDK 1.8：

注意事项

1. 目前服务端，不支持TLSv1.3，如果碰到 `[SSL: SSLV3_ALERT_HANDSHAKE_FAILURE] sslv3 alert handshake failure` 之类的 ssl 错误，请使用TLSv1 或TLSv1.1 或T LSv1.2来重新测试。
2. 目前服务端支持的 ssl 协议和加密套件如下，请选择合适的 ssl 版本：

```
ssl_protocols    TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers      AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
```

3. 服务地址和端口请参见 SMTP 服务地址 。

   以下是代码示例：

```java
package org.example;

import javax.mail.*;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.util.Properties;

public class SampleMail {
    private static final String SMTP_HOST = "smtp.qcloudmail.com";
    private static final String SMTP_PORT = "465";

    public static void main(String[] args) {
        // 配置发送邮件的环境属性
        final Properties props = new Properties();
        // 表示SMTP发送邮件，需要进行身份验证
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.host", SMTP_HOST);
        // 如果使用ssl，则去掉使用25端口的配置，进行如下配置,
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.socketFactory.port", SMTP_PORT);
        props.put("mail.smtp.port", SMTP_PORT);
        // 发件人的账号，填写控制台配置的发信地址,比如xxx@xxx.com
```

```
        props.put("mail.user", "xxx@xxx.com");
        // 访问SMTP服务时需要提供的密码(在控制台选择发信地址进行设置)
        props.put("mail.password", "XXXX");
        props.setProperty("mail.smtp.socketFactory.fallback", "false");
        props.put("mail.smtp.ssl.enable", "true");
        //props.put("mail.smtp.starttls.enable","true");
        // 构建授权信息，用于进行SMTP进行身份验证
        Authenticator authenticator = new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                // 用户名、密码
                String userName = props.getProperty("mail.user");
                String password = props.getProperty("mail.password");
                return new PasswordAuthentication(userName, password);
            }
        };
        // 使用环境属性和授权信息，创建邮件会话
        Session mailSession = Session.getInstance(props, authenticator);
//        mailSession.setDebug(true);
        //UUID uuid = UUID.randomUUID();
        //final String messageIDValue = "<" + uuid.toString() + ">";
        // 创建邮件消息
        MimeMessage message = new MimeMessage(mailSession) {
            //@Override
            //protected void updateMessageID() throws MessagingException {
            //设置自定义Message-ID值
            //setHeader("Message-ID", messageIDValue);
            //}
        };
        try {
            // 设置发件人邮件地址和名称。填写控制台配置的发信地址,比如xxx@xxx.com。和
上面的mail.user保持一致。名称用户可以自定义填写。
            InternetAddress from = new InternetAddress("xxx@xxx.com", "test");
            message.setFrom(from);
            //可选。设置回信地址
//            Address[] a = new Address[1];
//            a[0] = new InternetAddress("***");
//            message.setReplyTo(a);
            // 设置收件人邮件地址，比如yyy@yyy.com
            InternetAddress to = new InternetAddress("xxx@xxx.com");
            message.setRecipient(MimeMessage.RecipientType.TO, to);
            //如果同时发给多人，才将上面两行替换为如下（因为部分收信系统的一些限制，尽量
每次投递给一个人；同时我们限制单次允许发送的人数是50人）：
            //InternetAddress[] adds = new InternetAddress[2];
            //adds[0] = new InternetAddress("xxx@xxx.com");
```

```
            //adds[1] = new InternetAddress("xxx@xxx.com");
            //message.setRecipients(Message.RecipientType.TO, adds);

            // 设置邮件标题
            message.setSubject("测试邮件");
            message.setHeader("Content-Transfer-Encoding", "base64");
            // 设置邮件的内容体 type: text/plain（纯文本）text/html（HTML 文档）
            message.setContent("<!DOCTYPE html>\n<html>\n<head>\n<meta
charset=\"utf-8\">\n<title>hello world</title>\n</head>\n<body>\n " +
                    "<h1>我的第一个标题</h1>\n    <p>我的第一个段落。
</p>\n</body>\n</html>", "text/html;charset=UTF-8");
            //发送邮件
            Transport.send(message);
        } catch (MessagingException | UnsupportedEncodingException e) {
            String err = e.getMessage();
            err = new String(err.getBytes(StandardCharsets.ISO_8859_1),
StandardCharsets.UTF_8);
            System.out.println(err);
        }
    }

}
```

发送附件

```
package org.example;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.util.Properties;
import java.util.UUID;

public class SampleMailAttach {
    private static final String SMTP_HOST = "smtp.qcloudmail.com";
    private static final String SMTP_PORT = "465";

    public static void main(String[] args) {
        // 配置发送邮件的环境属性
        final Properties props = new Properties();
        // 表示SMTP发送邮件，需要进行身份验证
```

```
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.host", SMTP_HOST);
        // 如果使用ssl，则去掉使用25端口的配置，进行如下配置,
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.socketFactory.port", SMTP_PORT);
        props.put("mail.smtp.port", SMTP_PORT);
        // 发件人的账号，填写控制台配置的发信地址,比如xxx@xxx.com
        props.put("mail.user", "xxx@xxx.com");
        // 访问SMTP服务时需要提供的密码(在控制台选择发信地址进行设置)
        props.put("mail.password", "XXXX");
        props.setProperty("mail.smtp.socketFactory.fallback", "false");
        props.put("mail.smtp.ssl.enable", "true");
        //props.put("mail.smtp.starttls.enable","true");
        // 构建授权信息，用于进行SMTP进行身份验证
        Authenticator authenticator = new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                // 用户名、密码
                String userName = props.getProperty("mail.user");
                String password = props.getProperty("mail.password");
                return new PasswordAuthentication(userName, password);
            }
        };
        // 使用环境属性和授权信息，创建邮件会话
        Session mailSession = Session.getInstance(props, authenticator);

        UUID uuid = UUID.randomUUID();
        final String messageIDValue = "<" + uuid.toString() + ">";
        //创建邮件消息
        MimeMessage message = new MimeMessage(mailSession) {
            @Override
            protected void updateMessageID() throws MessagingException {
                //设置自定义Message-ID值
                setHeader("Message-ID", messageIDValue);
            }
        };
        try {
            // 设置发件人邮件地址和名称。填写控制台配置的发信地址,和mail.user保持一致。发信
别名可以自定义，如test。
            InternetAddress from = new InternetAddress("xxx@xxx.com", "test");
            message.setFrom(from);
            //可选。设置回信地址
            Address[] a = new Address[1];
            a[0] = new InternetAddress("xxx@xxx.com");
            message.setReplyTo(a);
```

```
        //设置收件人邮件地址，比如yyy@yyy.com
        InternetAddress to = new InternetAddress("xxx@xxx.com");
        message.setRecipient(MimeMessage.RecipientType.TO, to);
        //如果同时发给多人，才将上面两行替换为如下（因为部分收信系统的一些限制，尽量每次
投递给一个人；同时我们限制单次允许发送的人数是50人）：
        /*InternetAddress[] adds = new InternetAddress[2];
        adds[0] = new InternetAddress("xxx@xxx.com");
        adds[1] = new InternetAddress("xxx@xxx.com");
        message.setRecipients(Message.RecipientType.TO, adds);*/

        // 设置邮件标题
        message.setSubject("测试邮件");
        //发送附件，总的邮件大小不超过10M，创建消息部分
        BodyPart messageBodyPart = new MimeBodyPart();
        //消息 text/plain（纯文本）text/html（HTML 文档）
        messageBodyPart.setText("<!DOCTYPE html>\n<html>\n<head>\n<meta
charset=\"utf-8\">\n<title>hello world</title>\n</head>\n<body>\n " +
                "<h1>我的第一个标题</h1>\n    <p>我的第一个段落。
</p>\n</body>\n</html>");
        messageBodyPart.setHeader("Content-Type", "text/plain;charset=utf-8");
        //创建多重消息
        Multipart multipart = new MimeMultipart();
        //设置文本消息部分
        multipart.addBodyPart(messageBodyPart);
        //附件部分
        messageBodyPart = new MimeBodyPart();
        //设置要发送附件的文件路径
        String filename = "/Users/aaa/bbb/a.txt";
        FileDataSource source = new FileDataSource(filename);
        messageBodyPart.setDataHandler(new DataHandler(source));
        //处理附件名称中文（附带文件路径）乱码问题
        String filenameEncode = MimeUtility.encodeText(filename, "UTF-8", "base64");
        messageBodyPart.setFileName(filenameEncode);
        messageBodyPart.setHeader("Content-Transfer-Encoding", "base64");
        messageBodyPart.setHeader("Content-Disposition", "attachment");
        messageBodyPart.setHeader("Content-Type", "application/octet-
stream;name=\"" + filenameEncode + "\"");
        multipart.addBodyPart(messageBodyPart);

        //附件部分,多个附件，分为多个part
        BodyPart messageBodyPart1 = new MimeBodyPart();
        //设置要发送附件的文件路径
        String filename1 = "/Users/aaa/bbb/b.txt";
        FileDataSource source1 = new FileDataSource(filename1);
        messageBodyPart1.setDataHandler(new DataHandler(source1));
```

```
            //处理附件名称中文（附带文件路径）乱码问题
            String filenameEncode1 = MimeUtility.encodeText(filename1, "UTF-8",
"base64");
            messageBodyPart1.setHeader("Content-Transfer-Encoding", "base64");
            messageBodyPart1.setHeader("Content-Disposition", "attachment");
            messageBodyPart1.setHeader("Content-Type", "application/octet-
stream;name=\"" + filenameEncode1 + "\"");
            multipart.addBodyPart(messageBodyPart1);

            //发送含有附件的完整消息
            message.setContent(multipart);
            //发送附件代码，结束
            //发送邮件
            Transport.send(message);
        } catch (MessagingException | UnsupportedEncodingException e) {
            String err = e.getMessage();
            err = new String(err.getBytes(StandardCharsets.ISO_8859_1),
StandardCharsets.UTF_8);
            System.out.println(err);
        }
    }
}
```

常见问题

遇到报错：“No appropriate protocol (protocol is disabled or cipher suites are inappropriate)”，如何处理?

找到 `jdk/jre/lib/security/java.security` 文件进行修改：

```
#    jdk.tls.disabledAlgorithms=MD5, SSLv3, DSA, RSA keySize < 2048
#jdk.tls.disabledAlgorithms=SSLv3, RC4, MD5withRSA, DH keySize < 768
```

Go 调用示例

最近更新时间：2024-04-09 17:52:51

以下代码示例，Go 语言（版本为1.16）通过 SMTP 发送邮件：

注意事项

1. 目前服务端，不支持TLSv1.3，如果碰到 `[SSL: SSLV3_ALERT_HANDSHAKE_FAILURE] sslv3 alert handshake failure` 之类的 ssl 错误，请使用 TLSv1 或 TLSv1.1 或TLSv1.2来重新测试。
2. 目前服务端支持的 ssl 协议和加密套件如下，请选择适合的 ssl 版本：

```
ssl_protocols    TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers      AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
```

3. 服务地址和端口请参见 SMTP 服务地址 。

   以下是代码示例：

```go
package main

import (
    "crypto/tls"
    "fmt"
    "log"
    "net"
    "net/smtp"
)

// Test465  for port 465
func Test465() error {
    host := "smtp.qcloudmail.com"
    port := 465
    //控制台创建的发信地址
    email := "abc@cd.com"
    //控制台设置的SMTP密码
    password := "****"
    toEmail := "test@test123.com"
    header := make(map[string]string)
    header["From"] = "test " + "<" + email + ">"
    header["To"] = toEmail
    header["Subject"] = "test subject"
    //html格式邮件
```

```
    header["Content-Type"] = "text/html; charset=UTF-8"
    body := "<!DOCTYPE html>\n<html>\n<head>\n<meta charset=\"utf-
8\">\n<title>hello world</title>\n</head>\n<body>\n " +
        "<h1>我的第一个标题</h1>\n    <p>我的第一个段落。
</p>\n</body>\n</html>"
    //纯文本格式邮件
    //header["Content-Type"] = "text/plain; charset=UTF-8"
    //body := "test body"
    message := ""
    for k, v := range header {
        message += fmt.Sprintf("%s: %s\r\n", k, v)
    }
    message += "\r\n" + body
    auth := smtp.PlainAuth(
        "",
        email,
        password,
        host,
    )
    err := SendMailWithTLS(
        fmt.Sprintf("%s:%d", host, port),
        auth,
        email,
        []string{toEmail},
        []byte(message),
    )
    if err != nil {
        fmt.Println("Send email error:", err)
    } else {
        fmt.Println("Send mail success!")
    }
    return err
}

// Dial return a smtp client
func Dial(addr string) (*smtp.Client, error) {
    conn, err := tls.Dial("tcp", addr, nil)
    if err != nil {
        log.Println("tls.Dial Error:", err)
        return nil, err
    }

    host, _, _ := net.SplitHostPort(addr)
    return smtp.NewClient(conn, host)
}
```

```
// SendMailWithTLS send email with tls
func SendMailWithTLS(addr string, auth smtp.Auth, from string,
    to []string, msg []byte) (err error) {
    //create smtp client
    c, err := Dial(addr)
    if err != nil {
        log.Println("Create smtp client error:", err)
        return err
    }
    defer c.Close()
    if auth != nil {
        if ok, _ := c.Extension("AUTH"); ok {
            if err = c.Auth(auth); err != nil {
                log.Println("Error during AUTH", err)
                return err
            }
        }
    }
    if err = c.Mail(from); err != nil {
        return err
    }
    for _, addr := range to {
        if err = c.Rcpt(addr); err != nil {
            return err
        }
    }
    w, err := c.Data()
    if err != nil {
        return err
    }
    _, err = w.Write(msg)
    if err != nil {
        return err
    }
    err = w.Close()
    if err != nil {
        return err
    }
    return c.Quit()
}

func main() {
    Test465()
}
```

PHP 调用示例

最近更新时间：2024-04-09 17:52:51

注意事项

1. 推荐用户使用 PHPMailer 包：
   - 如果是新项目并且使用 composer 那么只需在 composer.json 加上 `"phpmailer/phpmailer": "^6.5"` ，或者执行 `composer require phpmailer/phpmailer` ，然后使用下面的代码即可。
   - 如果是老项目且没有使用 composer 的，需手动引入 PHPMailer 。
2. 目前服务端，不支持TLSv1.3，如果碰到 `[SSL: SSLV3_ALERT_HANDSHAKE_FAILURE] sslv3 alert handshake failure` 之类的 ssl 错误，请使用 TLSv1 或 TLSv1.1 或 TLSv1.2来重新测试。
3. 目前服务端支持的 ssl 协议和加密套件如下，请选择适合的 ssl 版本：

```
ssl_protocols    TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers      AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
```

4. 服务地址和端口请参见 SMTP 服务地址 。

   以下是代码示例：

```php
<?php

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;
require './PHPMailer/src/Exception.php';
require './PHPMailer/src/PHPMailer.php';
require './PHPMailer/src/SMTP.php';

$mail = new PHPMailer(true);

try {
    //Server settings
    $mail->SMTPDebug  = SMTP::DEBUG_SERVER;                 //Enable verbose
debug output
    $mail->SMTPAuth   = true;                               //Enable SMTP authentication
    //$mail->AuthType   = 'LOGIN';
    $mail->isSMTP();                                        //Send using SMTP
```

```
    $mail->Host       = 'smtp.qcloudmail.com';                //Set the SMTP server to
send through
    $mail->Username   = 'abc@qq.aa.com';          //SMTP username
    $mail->Password   = '123456';                 //SMTP password

    $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;           //Enable implicit
TLS encryption
    $mail->CharSet     = PHPMailer::CHARSET_UTF8;
    $mail->CharSet     = 'UTF-8';
    $mail->ContentType = 'text/plain; charset=UTF-8';
    $mail->Encoding    = PHPMailer::ENCODING_BASE64;
    //$mail->Encoding   = '8bit';
    $mail->Port       = 465;                              //TCP port to connect to; use 587 if
you have set `SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS`

    //Recipients
    $mail->setFrom('abc@qq.aa.com', 'fromName');
    $mail->addAddress('test@test.com', 'toName');    //Add a recipient
    //$mail->addAddress('ellen@example.com');                 //Name is optional
    //$mail->addReplyTo('info@example.com', 'Information');
    //$mail->addCC('cc@example.com');
    //$mail->addBCC('bcc@example.com');

    //Attachments
    $mail->addAttachment('./tmp.txt');                        //Add attachments
    //$mail->addAttachment('/tmp/image.jpg', 'new.jpg');        //Optional name

    //Content
    //$mail->isHTML(true);                                    //Set email format to HTML
    $mail->Subject = 'Here is the subject';
    $mail->Body    = 'This is the HTML message body <b>in bold!</b>';
    //$mail->AltBody = 'This is the body in plain text for non-HTML mail clients';

    $mail->send();
    echo 'Message has been sent';
} catch (Exception $e) {
    echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
}
```

Python 调用示例

最近更新时间：2023-10-12 14:39:31

以下代码示例，Python 语言（版本 >= 3.6即可）通过 SMTP 发送邮件：

```
# -*- coding:utf-8 -*-
import smtplib
import ssl
from email.header import Header
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.utils import formataddr


def send_mail(sender, sender_alias, sender_pwd, recipient_list, subject, body,
host="smtp.qcloudmail.com", port=465,
            is_use_ssl=True):
    try:
        message = MIMEMultipart('alternative')
        message['Subject'] = Header(subject, 'UTF-8')
        message['From'] = formataddr([sender_alias, sender])
        message['To'] = ",".join(recipient_list)
        to_addr_list = recipient_list

        mime_text = MIMEText(body, _subtype='html', _charset='UTF-8')
        message.attach(mime_text)

        if is_use_ssl:
            context = ssl.create_default_context()
            context.set_ciphers('DEFAULT')
            # context.options |= ssl.OP_NO_TLSv1_3  # 如果当前选择到了TLSv1_3，则使用这
行来屏蔽1.3因为服务端不支持

            # 当使用了上面代码还是出现 SSL: SSLV3_ALERT_HANDSHAKE_FAILURE] sslv3
alert handshake failure 之类的ssl错误，
            # 请先使用 pip install --upgrade ssl 来更新SSL库，重新运行，如果仍旧失败，则需要
更改下面ses的支持的协议和加密算法
            # 目前ses服务端支持ssl的协议和加密套件如下：
            # ssl_protocols    TLSv1 TLSv1.1 TLSv1.2;
            # ssl_ciphers      AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
            # 使用下面方法来自定义协议和加密算法
            # print(f'ssl_version={ssl.OPENSSL_VERSION}')
            # context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
```

```
        # context.set_ciphers('AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5')

        client = smtplib.SMTP_SSL(host, port, context=context)
    else:
        client = smtplib.SMTP(host, port)

    client.login(sender, sender_pwd)
    client.sendmail(sender, to_addr_list, message.as_string())
    client.quit()

    print('Send email success!')
except smtplib.SMTPConnectError as e:
    print('Send email failed,connection error:', e.smtp_code, e.smtp_error)
except smtplib.SMTPAuthenticationError as e:
    print('Send email failed,smtp authentication error:', e.smtp_code, e.smtp_error)
except smtplib.SMTPSenderRefused as e:
    print('Send email failed,sender refused:', e.smtp_code, e.smtp_error)
except smtplib.SMTPRecipientsRefused as e:
    print('Send email failed,recipients refused:', e.recipients)
except smtplib.SMTPDataError as e:
    print('Send email failed,smtp data error:', e.smtp_code, e.smtp_error)
except smtplib.SMTPException as e:
    print('Send email failed,smtp exception:', str(e))
except Exception as e:
    print('Send email failed,other error:', str(e))


if __name__ == '__main__':
    # 将下面xxx之类替换为自己的地址，直接运行就可以成功发送邮件了，python版本>=3.6即可
    # 控制台创建的发信人地址
    from_email = "xxx@xxxx"
    # 控制台发信地址对应设置的SMTP密码
    from_email_pwd = "xxx"
    # 接收人地址列表
    to_email_list = ["xxx@xxx1", "xxx@xxx2"]

    # 发信人地址别名
    from_alias = "测试别名"
    # 邮件主题
    subject_txt = "[测试主题]"
    # 邮件内容
    body_content = (
```

```
        "<!DOCTYPE html>\n<html>\n<head>\n<meta charset=\"utf-8\">\n<title>hello
world</title>\n</head>\n<body>\n "
        "<h1>我的第一个标题</h1>\n    <p>我的第一个段落。</p>\n</body>\n</html>")

    # 默认使用ssl发邮件，端口可以为 465或587
    is_using_ssl = True
    # smtp服务地址, 香港=smtp.qcloudmail.com,新加坡=sg-smtp.qcloudmail.com,广州
=gz-smtp.qcloudmail.com
    smtp_host = "smtp.qcloudmail.com"
    # smtp端口号, 465和587使用ssl加密，25使用tls
    smtp_port = 465

    # 使用25端口发邮件
    # is_using_ssl = False
    # smtp_host = "smtp.qcloudmail.com"
    # smtp_port = 25
    send_mail(from_email, from_alias, from_email_pwd, to_email_list, subject_txt,
body_content,
              smtp_host, smtp_port, is_using_ssl)
```

发送带附件的邮件

最近更新时间：2022-01-21 14:49:33

通过 SMTP 的方式发送带附件的邮件的方法，即构建一封 MIME 格式的邮件内容。

邮件 mime 格式

了解更多协议相关，请参见 MIME 协议 。

> **说明**
> MIME 消息由消息头和消息体两大部分组成。分别称为 邮件头 、 邮件体 。

邮件头

包含了发件人、收件人、主题、时间、MIME 版本、邮件内容的类型等重要信息。

> **说明**
> 每条信息称为一个域，由域名后加“:”和信息内容构成，可以是一行，较长的也可以占用多行。
> - 域的首行必须“顶头”写，即左边不能有空白字符（空格和制表符）。
> - 续行则必须以空白字符打头，且一个空白字符不是信息本身固有的（解码时要过滤掉）。

邮件头中不允许出现空行。有一些邮件不能被邮件客户端软件识别，显示的是原始码，就是因为首行是空行。
例如：

内容	示例
Date	Mon, 29 Jun 2009 18:39:03 +0800
From	abc@123.com
To	abc1@123.com
BCC	abc3@123.com
Subject	test
Message-ID	123@123.com
Mime-Version	1.0

域名	含义
Bcc	暗送地址

Cc	抄送地址
Content-Transfer-Encoding	内容的传输编码方式
Content-Type	内容的类型
Date	日期和时间
Delivered-To	发送地址
From	发件人地址
Message-ID	消息 ID
MIME-Version	MIME 版本
Received	传输路径
Reply-To	回复地址
Return-Path	回复地址
Subject	主题
To	收件人地址

邮件体

域名	含义
Content-ID	段体的 ID
Content-Transfer-Encoding	段体的传输编码方式
Content-Location	段体的位置（路径）
Content-Base	段体的基位置
Content-Disposition	段体的安排方式
Content-Type	段体的类型

有些域除了值之外，还带有参数。值与参数、参数与参数之间以“;”分隔。参数名与参数值之间以“=”分隔。

- 邮件体包含邮件的内容，它的类型由邮件头的 `Content-Type` 域指出。

> **说明**
> 常见的简单类型有：

- text/plain（纯文本）
- text/html（超文本）

- multipart 类型，是 MIME 邮件的精髓。邮件体被分为多个段，每个段又包含段头和段体两部分，这两部分之间也以空行分隔。
- 常见的 multipart 类型有三种：
- multipart/mixed
- multipart/related
- multipart/alternative

可从上述名称，得知这些类型各自的含义和用处。它们之间的层次关系可归纳为下图所示：

```
+------------------------- multipart/mixed ----------------------------+
|                                                                      |
|  +----------------- multipart/related ------------------+            |
|  |                                                      |            |
|  |  +----- multipart/alternative ------+  +----------+  |  +------+  |
|  |  |                                 |  | 内嵌资源 |  |  | 附件 |  |
|  |  |  +------------+  +------------+  |  +----------+  |  +------+  |
|  |  |  | 纯文本正文 |  | 超文本正文 |  |                |            |
|  |  |  +------------+  +------------+  |  +----------+  |  +------+  |
|  |  |                                 |  | 内嵌资源 |  |  | 附件 |  |
|  |  +---------------------------------+  +----------+  |  +------+  |
|  |                                                      |            |
|  +------------------------------------------------------+            |
|                                                                      |
+----------------------------------------------------------------------+
```

如果在邮件中要添加附件，必须定义 multipart/mixed 段；如果存在内嵌资源，至少要定义 multipart/related 段；如果纯文本与超文本共存，至少要定义 multipart/alternative 段。

> **说明**
> 附件个数不超过10个，单个附件大小不超过4M，总附件大小不超过8M，具体可参见 数据结构 。

代码示例

```
package main
import (
    "bytes"
    "crypto/tls"
```

```
    "encoding/base64"
    "fmt"
    "io/ioutil"
    "log"
    "mime"
    "net"
    "net/smtp"
    "time"
)

// Test465Attachment  for port 465
func Test465Attachment() error {
    boundary := "GoBoundary"
    host := "smtp.qcloudmail.com"
    port := 465
    email := "abc@cd.com"
    password := "***"
    toEmail := "test@test123.com"
    header := make(map[string]string)
    header["From"] = "test " + "<" + email + ">"
    header["To"] = toEmail
    header["Subject"] = "Test465Attachment"
    header["Content-Type"] = "multipart/mixed;boundary=" + boundary
    //该字段暂时没有用到，默认传1.0
    header["Mime-Version"] = "1.0"
    //该字段暂时没有用到
    header["Date"] = time.Now().String()
    bodyHtml := "<!DOCTYPE html>\n<html>\n<head>\n<meta charset=\"utf-
8\">\n<title>hello world</title>\n</head>\n<body>\n " +
        "<h1>我的第一个标题</h1>\n    <p>我的第一个段落。</p>\n</body>\n</html>"
    message := ""
    for k, v := range header {
        message += fmt.Sprintf("%s: %s\r\n", k, v)
    }
    buffer := bytes.NewBuffer(nil)
    buffer.WriteString(message)
    contentType := "Content-Type: text/html" + "; charset=UTF-8"
    body := "\r\n--" + boundary + "\r\n"
    body += "Content-Type:" + contentType + "\r\n"
    body += "\r\n" + bodyHtml + "\r\n"
    buffer.WriteString(body)

    attachment := "\r\n--" + boundary + "\r\n"
    attachment += "Content-Transfer-Encoding:base64\r\n"
    attachment += "Content-Disposition:attachment\r\n"
```

```
    attachment += "Content-Type:" + "application/octet-stream" + ";name=\"" +
mime.BEncoding.Encode("UTF-8",
        "./go.mod") + "\"\r\n"
    buffer.WriteString(attachment)
    writeFile(buffer, "./go.mod")
    //多个附件往后拼接，最多不能超过10个附件，单个附件不能超过5M,所有附件累计不能超过8-
9M，消息体过大会返回EOF
    attachment1 := "\r\n--" + boundary + "\r\n"
    attachment1 += "Content-Transfer-Encoding:base64\r\n"
    attachment1 += "Content-Disposition:attachment\r\n"
    attachment1 += "Content-Type:" + "application/octet-stream" + ";name=\"" +
mime.BEncoding.Encode("UTF-8",
        "./bbbb.txt") + "\"\r\n"
    buffer.WriteString(attachment1)
    writeFile(buffer, "./bbbb.txt")
    defer func() {
        if err := recover(); err != nil {
            log.Fatalln(err)
        }
    }()

    buffer.WriteString("\r\n--" + boundary + "--")
    message += "\r\n" + body
    auth := smtp.PlainAuth(
        "",
        email,
        password,
        host,
    )
    err := SendMailWithTLS(
        fmt.Sprintf("%s:%d", host, port),
        auth,
        email,
        []string{toEmail},
        buffer.Bytes(),
    )
    if err != nil {
        fmt.Println("Send email error:", err)
    } else {
        fmt.Println("Send mail success!")
    }
    return err
}

// Dial return a smtp client
```

```
func Dial(addr string) (*smtp.Client, error) {
    conn, err := tls.Dial("tcp", addr, nil)
    if err != nil {
        log.Println("tls.Dial Error:", err)
        return nil, err
    }

    host, _, _ := net.SplitHostPort(addr)
    return smtp.NewClient(conn, host)
}

// SendMailWithTLS send email with tls
func SendMailWithTLS(addr string, auth smtp.Auth, from string,
    to []string, msg []byte) (err error) {
    //create smtp client
    c, err := Dial(addr)
    if err != nil {
        log.Println("Create smtp client error:", err)
        return err
    }
    defer c.Close()
    if auth != nil {
        if ok, _ := c.Extension("AUTH"); ok {
            if err = c.Auth(auth); err != nil {
                log.Println("Error during AUTH", err)
                return err
            }
        }
    }
    if err = c.Mail(from); err != nil {
        return err
    }
    for _, addr := range to {
        if err = c.Rcpt(addr); err != nil {
            return err
        }
    }
    w, err := c.Data()
    if err != nil {
        return err
    }
    _, err = w.Write(msg)
    if err != nil {
        return err
    }
```

```
    err = w.Close()
    if err != nil {
        return err
    }
    return c.Quit()
}

// writeFile read file to buffer
func writeFile(buffer *bytes.Buffer, fileName string) {
    file, err := ioutil.ReadFile(fileName)
    if err != nil {
        panic(err.Error())
    }
    payload := make([]byte, base64.StdEncoding.EncodedLen(len(file)))
    base64.StdEncoding.Encode(payload, file)
    buffer.WriteString("\r\n")
    for index, line := 0, len(payload); index < line; index++ {
        buffer.WriteByte(payload[index])
        if (index+1)%76 == 0 {
            buffer.WriteString("\r\n")
        }
    }
}

func main() {
    Test465Attachment()
}
```

错误码

最近更新时间：2021-12-09 16:35:46

输入参数

域名	含义	备注
Bcc	暗送地址	目前不支持
Cc	抄送地址	目前不支持
Content-Transfer-Encoding	内容的传输编码方式	目前没有用到，不用传，除了附件内容之外，其他的内容无需加密
Content-Type	内容的类型	目前必须传 text/plain; charset=UTF-8,text/html; charset=UTF-8 multipart/mixed, multipart/related 和 multipart/alternative 中的一种，传其他的会报错
Date	日期和时间	目前没有用到
Delivered-To	发送地址	目前没有用到
From	发件人地址	必传
Message-ID	消息 ID	目前没有用到
MIME-Version	MIME 版本	目前没有用到，不传或者传1.0。否则会报错
Received	传输路径	目前没有用到
Reply-To	回复地址	目前没有用到
Return-Path	回复地址	目前没有用到
Subject	主题	必传
To	收件人地址	必传

附件部分参数（发送附件时）

域名	含义	备注
Content-Type	段体的类型	文件建议传 application/octet-stream
Content-	段体的传输编	目前仅支持传 base64，传其他的会报错

Transfer-Encoding	码方式	
Content-Disposition	段体的安排方式	目前仅支持传 attachment，传其他的会导致无法发送附件
Content-ID	段体的 ID	目前不支持
Content-Location	段体的位置(路径)	目前不支持
Content-Base	段体的基位置	目前不支持

说明
输入参数校验要求总体与 发信接口 保持一致：包含收件人邮箱数量，邮件正文大小，附件格式，附件大小等限制。

返回参数

SMTP 接口无返回参数，仅支持返回 err 信息。若返回 nil，则标识调用接口成功，但实际发信不一定成功，获取邮件的发送状态可参见 获取邮件发送的状态 。

错误码

系统级错误

1. body 部分单行超过2000，或以上更多
   `554 5.0.0 Error: transaction failed, blame it on the weather: smtp: too longer line in input stream`
   或其他包含 `too longer` 的日志。
   `write tcp *.*.*.*:60575->*.*.*.*:25: write: broken pipe`
2. 附件大小过大
   附件9M左右 返回 EOF，建议附件总大小低于8M，整体报文大小不能超过10M，否则内容会截断，会报base64 解码失败等其他异常错误。

业务级错误

业务报错形式如下：

```
554 5.0.0 Error: transaction failed, blame it on the weather: ##SES-response-json: {"Response":{"RequestId":"bee4e9fb-8127-48cc-b606-bbb1e801596b","QcloudError":{"Error":{"Code":"FailedOperation.MissingEmailContent.操作失败。缺少发信内容（TemplateData和Simple不能同时为空)。
```

`##SES-response-json:` 之后的是发信接口返回的结构体的 `json` 形式，字段说明如下：

字段	字段类型	含义
RequestId	String	请求 LD
QcloudError	Stuct	错误结构体

QcloudError:

字段	字段类型	含义
Code	String	错误码
Message	String	错误信息

一般业务错误说明：

错误码	错误描述	备注
FailedOperation	msg.From is null	发信人为空
FailedOperation	msg.Subject is null	发信主题为空
FailedOperation	msg.Body is null	发信内容为空
FailedOperation	Content-Transfer-Encoding must in...	检查附件的 Content-Transfer-Encoding，参照入参说明
FailedOperation	Content-Type must in...	检查 Header 中的 Content-Type，参照入参说明
FailedOperation	Mime-Version must in...	检查 Header 中的 Mime-Version，参照入参说明
FailedOperation	The email is too large. Remove some content...	除了附件之外的邮件正文不能超过1M
FailedOperation	Incorrect attachment content. Make sure the base64 content is...	附件的内容需要 base64加密
FailedOperation	The attachments are too large. Make sure they do not exceed the...	单个附件超过5M。或所有附件大小超过10M（具体大小可能会调整）
RequestLimitExceeded.SmtpRateLimit	smtp sending frequency limit...	触发 SMTP 调用频率限制

其他业务报错：

您可参见 发送邮件 中错误码描述。