**汇总相关资料**

libcurl ===========》数据传输调用库

SMTP协议 & MIME协议 ===========》邮件发送协议

POP3协议 & IMAP协议 ===========》邮件接收协议

Telnet ===========》测试邮箱服务器工具

smtp://mail.dev.appeon.net:587 ===========》内部邮箱服务器地址&端口

smtp://smtp.gmail.com:587 ===========》Google邮箱服务器地址&端口

smtp://smtp-mail.outlook.com:587 ===========》outlook邮箱服务器地址&端口

==========================================================================================================================

**邮箱服务器搭建：**（windows 10系统搭建服务器比较麻烦，因此使用公司邮箱服务器进行测试）

通常，Windows 10、Windows 7 等客户端操作系统不包含 SMTP 服务器功能。您可以使用 IIS 中的SMTP 电子邮件功能连接到 Windows Server 上的现有 SMTP 服务器。SMTP 电子邮件功能不是 SQL Server 数据库邮件所必需的 SMTP 服务器。因此，本主题不包括在客户端操作系统上安装和配置 SMTP 服务器的步骤。

MSDN：

<https://docs.microsoft.com/en-us/biztalk/install-and-config-guides/appendix-d-create-the-smtp-server>

==========================================================================================================================

**测试邮箱Exchange服务器方式：**(先安装Telnet)

通过Telnet进行邮箱服务器测试。

MSDN：

<https://docs.microsoft.com/en-us/exchange/mail-flow/test-smtp-with-telnet?view=exchserver-2019>

相关数据：

目标SMTP邮箱服务器：mail.dev.appeon.net:25

域名domain：dev.appeon.net

发件人：[zenzhaofa@dev.appeon.net](mailto:zenzhaofa@dev.appeon.net)

收件人：@dev.appeon.net

步骤：

1. cmd---------》telnet
2. set localecho
3. set logfile SMTPlog
4. open mail.dev.appeon.net:25
5. EHLO dev.appeon.net
6. mail from:zengzhaofa@dev.appeon.net
7. rcpt to:zengzhaofa@dev.appeon.net 🡪enter
8. data ==🡺enter
9. subject: test=🡺enter =🡺enter
10. use telnet test SMTP from cmd =🡺enter
11. .=🡺enter
12. quit

==========================================================================================================================

**发送邮件的三种方式：**

1. **加密发送邮件（SMTP+SSL/TLS）**
2. **不加密发送邮件（SMTP）**
3. **发送邮件带附件（MIME+SSL/TLS）**

==========================================================================================================================

**发送邮件的流程：**

1. **建立连接**
2. **发送邮件**
3. **断开连接**

**详细流程：**

* Open connection
* Authenticate
* Hello
* Authenticate
* From
* RcptTo
* Data
* Quit

1. 建立连接

当Client编辑好邮件后，点击发送的过程中，Client会和邮箱服务器建立连接。在进行连接过程中可以进行（SSL/TLS）协议加密，也可以进行邮箱服务器的证书认证。证书认证可以使用本地证书/自带证书文件。

1. 发送邮件

发送邮件过程中，不是一次性将数据传输到服务器，而是响应应答的方式。（具体看案例分析）。

1. 断开连接

断开连接过程需要释放libcurl资源。

**案例分析（Client和Server 应答流程）**

== Info: Trying 74.125.192.109:587...

== Info: Connected to smtp.gmail.com (74.125.192.109) port 587 (#0)

<= Recv header, 0000000054 bytes (0x00000036)

0000: 220 smtp.gmail.com ESMTP x16sm18277003qko.17 - gsmtp

=> Send header, 0000000022 bytes (0x00000016)

0000: EHLO developertestse

<= Recv header, 0000000052 bytes (0x00000034)

0000: 250-smtp.gmail.com at your service, [20.84.124.50]

<= Recv header, 0000000019 bytes (0x00000013)

0000: 250-SIZE 35882577

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250-8BITMIME

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250-STARTTLS

<= Recv header, 0000000025 bytes (0x00000019)

0000: 250-ENHANCEDSTATUSCODES

<= Recv header, 0000000016 bytes (0x00000010)

0000: 250-PIPELINING

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250-CHUNKING

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250 SMTPUTF8

=> Send header, 0000000010 bytes (0x0000000a)

0000: STARTTLS

<= Recv header, 0000000030 bytes (0x0000001e)

0000: 220 2.0.0 Ready to start TLS

== Info: TLSv1.0 (OUT), TLS header, Certificate Status (22):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: .....

== Info: TLSv1.3 (OUT), TLS handshake, Client hello (1):

=> Send SSL data, 0000000512 bytes (0x00000200)

0000: .......$r.F:l....Y..]..gDL..OyB.<.'.5. .\_..C^.S.... .=....0}/...

0040: ..,|.a..>.......,.0.........+./...$.(.k.#.'.g.....9.....3.....=.

0080: <.5./.....u.........smtp.gmail.com..............................

00c0: .............1.....\*.(.........................................+

0100: ............-.....3.&.$... }s.........\_.........K%.l..p..#W.....

0140: ................................................................

0180: ................................................................

01c0: ................................................................

== Info: TLSv1.2 (IN), TLS header, Certificate Status (22):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....z

== Info: TLSv1.3 (IN), TLS handshake, Server hello (2):

<= Recv SSL data, 0000000122 bytes (0x0000007a)

0000: ...v...nA....`...v..oH...KD.y..[D..qY. .\_..C^.S.... .=....0}/...

0040: ..,|.a.......3.$... G$..z..b/..q........F..:n\o..x.^.+....

== Info: TLSv1.2 (IN), TLS header, Finished (20):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: .....

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....;

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

== Info: TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):

<= Recv SSL data, 0000000006 bytes (0x00000006)

0000: ......

== Info: TLSv1.3 (IN), TLS handshake, Certificate (11):

<= Recv SSL data, 0000004002 bytes (0x00000fa2)

0000: ...........0...0..o.......j...f+.3...../.|0...\*.H........0F1.0..

0040: .U....US1"0 ..U....Google Trust Services LLC1.0...U....GTS CA 1C

0080: 30...220117030736Z..220411030735Z0.1.0...U....smtp.gmail.com0Y0.

00c0: ..\*.H.=....\*.H.=....B...T.$.]zA..].3O7\_"b1....!...x.ai.:....-I.>

0100: ....?]...h\_a..n..+...S....g0..c0...U...........0...U.%..0...+...

0140: ....0...U.......0.0...U......0....C\...b@....9...0...U.#..0....t

0180: ......=...F..q5.'0j..+........^0\0'..+.....0...http://ocsp.pki.

01c0: goog/gts1c301..+.....0..%http://pki.goog/repo/certs/gts1c3.der0.

0200: ..U....0...smtp.gmail.com0!..U. ..0.0...g.....0...+.....y...0<..

0240: U...50301./.-.+http://crls.pki.goog/gts1c3/moVDfISia2k.crl0.....

0280: +.....y............v.Q.....y.Vm.7x...z..'....B..........~f8.@...

02c0: ..G0E.!..I|7..J.rk.n..^.......M...>...... H|.Y......".....V....

0300: I...y....].v.F.U.u.. 0...i..}.,At..I.....p.mG...~f8.h.....G0E. .

0340: X....|. .....c.......p....u..|..!..$L.5.........q.......i.I.w.@1

0380: Y.0...\*.H.............$....}.I{S....9...4j.Sv...@.P.k..}.M..{1L.

03c0: .J..ZMZ..[..~|H.....UB.x..g.D"..F...{Wd.0..7.n.(c9.#Y...Re.8.].D

0400: ..m..]\.-,...Z......V...Fl)g..@..!.Otl.U...R...sv....M..UJ@....Q

0440: .G.v..Gp.()..Yx..w..Z...7O.;,..../f....)-...T1y..Nlx(.j...0.}+V.

0480: .3-(.i.......B.$....C......0...0..~..........SYk4....Pf0...\*.H..

04c0: ......0G1.0...U....US1"0 ..U....Google Trust Services LLC1.0...U

0500: ....GTS Root R10...200813000042Z..270930000042Z0F1.0...U....US1"

0540: 0 ..U....Google Trust Services LLC1.0...U....GTS CA 1C30.."0...\*

0580: .H.............0............b..7.7B.l...e.%...k..m.Z#.........|

05c0: ....B.^V$.z3....i..t.WLfh.w7US.9.M.4.\_%w7;...<......C...G..D.c..

0600: A..A0H......E.!..B...+eV4.&....}....H|7M?.....u..yW\.Wn.........

0640: %..,...\*....c.<PI....\_.+Y.....Q..w....O.pI.\m .......w.-...k...

0680: .+........'....Q.................0..|0...U...........0...U.%..0.

06c0: ..+.........+.......0...U.......0.......0...U.......t......=...

0700: F..q5.'0...U.#..0.....+&q.+H'./Rf,....q>0h..+........\0Z0&..+...

0740: ..0...http://ocsp.pki.goog/gtsr100..+.....0..$http://pki.goog/re

0780: po/certs/gtsr1.der04..U...-0+0).'.%.#http://crl.pki.goog/gtsr1/g

07c0: tsr1.crl0W..U. .P0N08..+.....y...0\*0(..+.........https://pki.goo

0800: g/repository/0...g.....0...g.....0...\*.H..............}. \.<...W

0840: .......rq.6...@..L.F...$..Pq"...n..jo......\_.l.......b....[.f..

0880: .......i>z.FI\_F.A...Me4...?O.lI..SA..!.....D[\*P..M.S6.B..T..wS.

08c0: d8'...X..|9-[..........S$....y.&.a.SR.B..f+?...........q.5($....

0900: .-.H.=Y.Q.t..|...[..4..........."....q....s$.7S...?..\.6..;.)...

0940: :b;lc...Yq.c'.L....s..\*....l2.3...Qq.4...].QX......Y.q..M(..m...

0980: ...F...k.w.....#.........D..u#.4.. ..^...RF.....!pQ.....U.+.3w.K

09c0: B..w..s.....7?..\*f.s.2.2l2....#.[}Mep.+.=...m.2.....c...]...q^\*.

0a00: .."..e:...e.....[.Y.G.-.$:...&....7..o....Q......Q......f0..b0.

0a40: .J.......w..l.6...!...X..0...\*.H........0W1.0...U....BE1.0...U..

0a80: ..GlobalSign nv-sa1.0...U....Root CA1.0...U....GlobalSign Root C

0ac0: A0...200619000042Z..280128000042Z0G1.0...U....US1"0 ..U....Googl

0b00: e Trust Services LLC1.0...U....GTS Root R10.."0...\*.H...........

0b40: ..0...............w.;...>...@<....}2..q.......j.....K.+........

0b80: ...........^..R..#'....c...~..^.h...ZG.M.3.N.....lK......d)%#...

0bc0: .=.`.......H.M..z.....Y........1.......ml....~&.E.=.y..(..&....

0c00: ..<h.S..:.+.....z..u....Vd..Oh.=......@..\....5l..P..L... .3.R.

0c40: .2.).%\*.H.r..d...........8f..c...x.{\w.v......y.W..&............

0c80: ..U.....K)...2%N\*.eD.....I...|..@{.C..l..}...L......K.....E.v..@

0cc0: +.S....;......1..w.o{>...".....2..c.Qr.]....)h3.:f...&...Wex'.^I

0d00: ......!............lH<@.~.Z.V<.....K.9K..?.U.n$..q..........A..

0d40: .=:..z.7...........80..40...U...........0...U.......0....0...U..

0d80: ......+&q.+H'./Rf,....q>0...U.#..0...`{f.E....P/}..4....K0`..+..

0dc0: ......T0R0%..+.....0...http://ocsp.pki.goog/gsr10)..+.....0...ht

0e00: tp://pki.goog/gsr1/gsr1.crt02..U...+0)0'.%.#.!http://crl.pki.goo

0e40: g/gsr1/gsr1.crl0;..U. .4020...g.....0...g.....0...+.....y....0..

0e80: .+.....y....0...\*.H.............4...(...v..1z!..R>..t.A..=5.....

0ec0: .\\_...|......W.&o[..Fh.7okz...7.%Q..h...I.Z...#...+.....Ij.u....

0f00: ...XHW.5.....o..o.......\*..Ni..-.h..+s....".7..f.I..U.g.2..&.p.=

0f40: .gm=|.4..2..n.jo.....K.;..7..D.~.l..F.....!.f...Ul.)...f[.wIH(..

0f80: ..3rS..5.b..$...9..~\*A.R.......?..

== Info: TLSv1.3 (IN), TLS handshake, CERT verify (15):

<= Recv SSL data, 0000000078 bytes (0x0000004e)

0000: ...J...F0D. !f...R...H.-.S..W.L.....=.z/T..i. td..e(..=....}.:`e

0040: Te...v9n.xe.V.

== Info: TLSv1.3 (IN), TLS handshake, Finished (20):

<= Recv SSL data, 0000000052 bytes (0x00000034)

0000: ...0>.......D...B..|R....x:...^.Q....e..j....\*;....~

== Info: TLSv1.2 (OUT), TLS header, Finished (20):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: .....

== Info: TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ....E

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

== Info: TLSv1.3 (OUT), TLS handshake, Finished (20):

=> Send SSL data, 0000000052 bytes (0x00000034)

0000: ...0.<.b`2..x;.....F....hdR^.b.......bT.)..w...`.q."

== Info: SSL connection using TLSv1.3 / TLS\_AES\_256\_GCM\_SHA384

== Info: Server certificate:

== Info: subject: CN=smtp.gmail.com

== Info: start date: Jan 17 03:07:36 2022 GMT

== Info: expire date: Apr 11 03:07:35 2022 GMT

== Info: issuer: C=US; O=Google Trust Services LLC; CN=GTS CA 1C3

== Info: SSL certificate verify result: unable to get local issuer certificate (20), continuing anyway.

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ....'

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000022 bytes (0x00000016)

0000: EHLO developertestse

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: .....

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

== Info: TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):

<= Recv SSL data, 0000000258 bytes (0x00000102)

0000: ........$R.}.....%..T~..9..9...9D.G.........+[......6..q..m0.t42

0040: .J4W..{A...eS.o.-..!Z7.....b@.9..;aD.`F..f...f......4H.....{z...

0080: .....n.N0....&./...6RA... ...U9x..V...F......|.......4Q[.q...R4.

00c0: ... .g7\...B...m.B..0k..3m.@..@..i".Q.....S ...;.%.....@.@....JJ

0100: ..

== Info: TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):

<= Recv SSL data, 0000000258 bytes (0x00000102)

0000: ........n..2.....%..T~..9..9...9....L[.....x.......(...-...K..{$

0040: b./....K.....V.."+@X"..j$K..U...eu.m..Rg........\....f#.X9@F.4.

0080: g.S1..K..............%=p1.W..,..B}.ZF....I../.k.%.\_~.......J.}.

00c0: ..i.....B......X.2./<#.#.........eQ]!..}#....}.z`k=J...X`.....JJ

0100: ..

== Info: old SSL session ID is stale, removing

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: .....

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000052 bytes (0x00000034)

0000: 250-smtp.gmail.com at your service, [20.84.124.50]

<= Recv header, 0000000019 bytes (0x00000013)

0000: 250-SIZE 35882577

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250-8BITMIME

<= Recv header, 0000000067 bytes (0x00000043)

0000: 250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUT

0040: H

<= Recv header, 0000000025 bytes (0x00000019)

0000: 250-ENHANCEDSTATUSCODES

<= Recv header, 0000000016 bytes (0x00000010)

0000: 250-PIPELINING

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250-CHUNKING

<= Recv header, 0000000014 bytes (0x0000000e)

0000: 250 SMTPUTF8

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: .....

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000012 bytes (0x0000000c)

0000: AUTH PLAIN

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: .....

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000006 bytes (0x00000006)

0000: 334

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ....C

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000050 bytes (0x00000032)

0000: AGlkYW9iYS56ZW5nQGdtYWlsLmNvbQAxMjM0NTZxYXouLkA=

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....%

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000020 bytes (0x00000014)

0000: 235 2.7.0 Accepted

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ....4

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000035 bytes (0x00000023)

0000: MAIL FROM:<idaoba.zeng@gmail.com>

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....;

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000042 bytes (0x0000002a)

0000: 250 2.1.0 OK x16sm18277003qko.17 - gsmtp

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ..../

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000030 bytes (0x0000001e)

0000: RCPT TO:<idaoba@outlook.com>

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....;

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000042 bytes (0x0000002a)

0000: 250 2.1.5 OK x16sm18277003qko.17 - gsmtp

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ....2

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000033 bytes (0x00000021)

0000: RCPT TO:<idaoba.zeng@gmail.com>

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....;

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000042 bytes (0x0000002a)

0000: 250 2.1.5 OK x16sm18277003qko.17 - gsmtp

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: .....

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send header, 0000000006 bytes (0x00000006)

0000: DATA

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....<

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000043 bytes (0x0000002b)

0000: 354 Go ahead x16sm18277003qko.17 - gsmtp

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: ....K

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

=> Send data, 0000000314 bytes (0x0000013a)

0000: Date: Mon, 29 Nov 2010 21:54:29 +1100

0027: To: <idaoba@outlook.com>

0041: From: <idaoba.zeng@gmail.com>

0060: Message-ID: <dcd7cb36-11db-487a-9f3a-e652a9458efd@dev.appeon.net

00a0: >

00a3: Subject: SMTP example message

00c2:

00c4: The body of the message starts here.

00ea:

00ec: It could be a lot of lines, could be MIME encoded, whatever.

012a: Check RFC5322.

== Info: TLSv1.2 (OUT), TLS header, Supplemental data (23):

=> Send SSL data, 0000000005 bytes (0x00000005)

0000: .....

=> Send SSL data, 0000000001 bytes (0x00000001)

0000: .

== Info: TLSv1.2 (IN), TLS header, Supplemental data (23):

<= Recv SSL data, 0000000005 bytes (0x00000005)

0000: ....G

<= Recv SSL data, 0000000001 bytes (0x00000001)

0000: .

<= Recv header, 0000000054 bytes (0x00000036)

0000: 250 2.0.0 OK 1644977065 x16sm18277003qko.17 - gsmtp

== Info: Connection #0 to host smtp.gmail.com left intact

==========================================================================================================================

1. **普通邮件流程**

curl = curl\_easy\_init();

if(curl)

{

curl\_easy\_setopt(curl, CURLOPT\_URL, "smtp://mail.example.com");

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_FROM, FROM\_ADDR);

recipients = curl\_slist\_append(recipients, TO\_ADDR);

recipients = curl\_slist\_append(recipients, CC\_ADDR);

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_RCPT, recipients);

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, payload\_source);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &upload\_ctx);

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

/\* Send the message \*/

res = curl\_easy\_perform(curl);

/\* 释放资源 \*/

curl\_slist\_free\_all(recipients);

curl\_easy\_cleanup(curl);

}

1. **SSL加密邮件流程(我们build的libcurl版本已经将SSL禁用)**

curl = curl\_easy\_init();

if(curl)

{

curl\_easy\_setopt(curl, CURLOPT\_USERNAME, "user"); // 用户名

curl\_easy\_setopt(curl, CURLOPT\_PASSWORD, "secret"); // 密码

// 邮箱服务器地址

curl\_easy\_setopt(curl, CURLOPT\_URL, "smtps://mainserver.example.net");

#ifdef SKIP\_PEER\_VERIFICATION

// 默认为认证服务器证书

curl\_easy\_setopt(curl, CURLOPT\_SSL\_VERIFYPEER, 0L);

#endif

#ifdef SKIP\_HOSTNAME\_VERIFICATION

// 默认为验证主机

curl\_easy\_setopt(curl, CURLOPT\_SSL\_VERIFYHOST, 0L);

#endif

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_FROM, FROM\_MAIL);

recipients = curl\_slist\_append(recipients, TO\_MAIL);

recipients = curl\_slist\_append(recipients, CC\_MAIL);

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_RCPT, recipients);

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, payload\_source);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &upload\_ctx);

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

curl\_easy\_setopt(curl, CURLOPT\_VERBOSE, 1L);

/\* Send the message \*/

res = curl\_easy\_perform(curl);

/\* Always cleanup \*/

curl\_slist\_free\_all(recipients);

curl\_easy\_cleanup(curl);

}

1. **TLS加密邮件流程**

curl = curl\_easy\_init();

if(curl)

{

/\* Set username and password \*/

curl\_easy\_setopt(curl, CURLOPT\_USERNAME, "user");

curl\_easy\_setopt(curl, CURLOPT\_PASSWORD, "secret");

curl\_easy\_setopt(curl, CURLOPT\_URL, "smtp://mainserver.example.net:587");

curl\_easy\_setopt(curl, CURLOPT\_USE\_SSL, (long)CURLUSESSL\_ALL);

/\* 自带证书 \*/

curl\_easy\_setopt(curl, CURLOPT\_CAINFO, "/path/to/certificate.pem");

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_FROM, FROM\_MAIL);

recipients = curl\_slist\_append(recipients, TO\_MAIL);

recipients = curl\_slist\_append(recipients, CC\_MAIL);

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_RCPT, recipients);

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, payload\_source);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &upload\_ctx);

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

curl\_easy\_setopt(curl, CURLOPT\_VERBOSE, 1L);

/\* Send the message \*/

res = curl\_easy\_perform(curl);

/\* Free the list of recipients \*/

curl\_slist\_free\_all(recipients);

/\* Always cleanup \*/

curl\_easy\_cleanup(curl);

}

1. **异步邮件流程**

curl\_global\_init(CURL\_GLOBAL\_DEFAULT);

curl = curl\_easy\_init();

if(!curl)

return 1;

mcurl = curl\_multi\_init();

if(!mcurl)

return 2;

curl\_easy\_setopt(curl, CURLOPT\_URL, "smtp://mail.example.com");

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_FROM, FROM\_MAIL);

recipients = curl\_slist\_append(recipients, TO\_MAIL);

recipients = curl\_slist\_append(recipients, CC\_MAIL);

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_RCPT, recipients);

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, payload\_source);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &upload\_ctx);

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

/\* Tell the multi stack about our easy handle \*/

curl\_multi\_add\_handle(mcurl, curl);

do

{

CURLMcode mc = curl\_multi\_perform(mcurl, &still\_running);

if(still\_running)

/\* wait for activity, timeout or "nothing" \*/

mc = curl\_multi\_poll(mcurl, NULL, 0, 1000, NULL);

if(mc)

break;

} while(still\_running);

/\* Free the list of recipients \*/

curl\_slist\_free\_all(recipients);

/\* Always cleanup \*/

curl\_multi\_remove\_handle(mcurl, curl);

curl\_multi\_cleanup(mcurl);

curl\_easy\_cleanup(curl);

curl\_global\_cleanup();

==========================================================================================================================

**LibCurl连接复用原理(libcurl库中的TCP连接复用部分)**

TCP复用流程：

curl = curl\_easy\_init();

while（1）

{

        // 设置相关信息

curl\_easy\_setopt();

        // 发送

        curl\_easy\_perform(curl);

        // 释放相关资源

// 重置

        curl\_easy\_reset(curl);

）

curl\_easy\_cleanup(curl);

**CURLOPT\_DNS\_CACHE\_TIMEOUT(缓存超时)**

curl\_easy\_setopt(CURL \*handle, CURLOPT\_DNS\_CACHE\_TIMEOUT, long age);

第三个参数：

默认：libcurl 将此信息缓存 60 秒。

0 ： libcurl 完全禁用缓存信息。

-1： libcurl 缓存的信息永久保存。

**CURLOPT\_FORBID\_REUSE(连接重用)**

curl\_easy\_setopt(CURL \*handle, CURLOPT\_FORBID\_REUSE, long close);

第三个参数：

默认：为0，使libcurl保持连接打开以供以后使用。

1： 使libcurl完成传输后显式关闭连接。

**1.连接池**

Libcurl针对tcp连接采用了连接池管理，一次传输完成后，它将在“连接池”（有时也称为连接缓存）中保持N个连接处于活动状态，以便恰好能够重用现有连接之一的后续传输可以使用它而不是创建一个新连接。

重用一个连接而不是创建一个新的连接在速度和所需资源方面提供了显著的好处。

当libcurl准备建立一个新的连接来进行传输时，它首先会检查池中是否有可以重用的现有连接。**连接重用检查是在使用任何DNS或其他名称解析机制之前完成的，因此它完全基于主机名。**如果已经存在到正确主机名的实时连接，则还将检查许多其他属性（端口号，协议等），以确保可以使用它。

**2.连接池场景**

Libcurl 几乎在所有的实现场景中都自动加入了连接池支持，主要的场景包括以下三种：

1. **Easy API pool**
2. **Multi API pool**
3. **Sharing the "connection cache"**

**Easy API pool**

当您使用easy API，或更具体地说，使用curl\_easy\_perform()时，libcurl将使该池与特定的easy句柄关联。 然后重用同一简单句柄将确保它可以重用其连接。

**Multi API pool**

当您使用multi API时，连接池将与multi句柄相关联。这允许您自由地清理和重新创建easy句柄，而不会有丢失连接池的风险，并且允许一个easy句柄使用的连接在以后的传输中被另一个简单句柄重用。只需重用multi句柄。

**Sharing the "connection cache"**

从libcurl 7.57.0开始，应用程序可以使用 [share interface]()，以使其他独立的传输共享同一连接池。

**3.Curl-TCP长连接和DNS解析的关系**

libcurl具有自己的内部DNS缓存，默认情况下它将在其中缓存解析的地址60秒（此选项可以更改） 。因此具有相同名称的后续解析将使用该时间范围内的缓存结果。

curl的连接缓存完全基于URL中使用的主机名，因此，如果缓存中已有与“ example.com”的可用连接，则该连接将用于对同一主机名的后续请求。 curl既不知道也不关心该名称的IP地址是什么，或者自连接启动以来它是否更改。 重用连接时，它将跳过整个dns解析阶段。

传输完成且连接仍然处于活动状态时，将把连接放回连接缓存中(或者，如果由于达到了限制而认为缓存已满，则关闭连接)。

由于连接重用是基于名称完成的，因此使用另一个名称解析为现有连接的相同IP不会使curl重复使用该连接。 它将解析名称并为此创建一个新的连接。

一个连接可以无限期地保留在连接缓存中，除非它被杀死以腾出空间或被重用。 如果它“死了”（由于它从另一端关闭），则当它被注意到时，它将最终从缓存中删除。

**HTTP/2**

可以通过HTTP/2发送的PING帧不会在连接缓存中处理(atm)连接，这将导致它们很快被服务器杀死。(libcurl 7.62.0添加了一个新的API，允许应用程序保持这样的连接，参见curl\_easy\_upkeep)

**DoH**

随着curl 7.62.0引入了DoH (dn -over- https)支持，DNS缓存将缓存TTL秒数的名称，而不仅仅是使用默认的60秒。

**4.重点总结**

Libcurl连接重用检查是在使用任何DNS或其他名称解析机制之前完成的，因此它完全基于主机名。当libcurl的池中已经有了到主机名的活动连接时，它会跳过名称解析，而是直接进行重用。

libcurl长连接 和 dns请求没有相关关系，因此原生的libcurl库并不能对于dns解析变化有很好的的感知过程。

**5. LibCurl 和 DNS解析的结合方案**

由于从上面的内容可以看出，LibCurl和DNS解析之间没有动态感应的过程，所以如果需要增加DNS动态感应过程不能从LibCurl下手，而应该通过LibCurl的调用方增加一定的策略来下手，主要的策略包括以下三种:

**DNS TTL方案：** 针对LibCurl库调用包装一层，增加一层域名的DNS TTL读取功能，根据TTL进行DNS请求定时器的设计，从而感应DNS的变化过程。但是需要注意的是TTL数据在标准的POSIX DNS解析 API中不可用，可以使用非dns完成名称解析。

**Libcurl Handle 增强控制方案**：Libcurl库对外统一暴露了handle内存区，为了感应到dns变化，我们就必须要让handle强制进入到dns解析阶段，因此可以针对Libcurl handle内存区的**存活时长** 和 内存区域**使用次数**下手，当时间超过了最大存活时长 或者 handle使用次数超过了最大复用次数，则强行进行Libcurl-handle内存区域释放，从而促使Libcurl进入dns解析过程。

**6.参考资料：**

* <https://ec.haxx.se/libcurl/libcurl-connectionreuse>
* <https://stackoverflow.com/questions/52940163/internal-connection-management-in-libcurl-and-dns-ttl/52944414>
* <https://curl.haxx.se/mail/lib-2017-06/0022.html>
* <https://curl.haxx.se/mail/lib-2016-10/0055.html>

==========================================================================================================================

**Debug日志相关**

**在libcurl中，有两种方式获取信息。**

1. curl\_easy\_getinfo - 从 curl 句柄中提取信息

2.开启Debug模式 - 从debug回调接口中将输出信息拷贝复制

**1.Debug相关定义和使用方式：**

typedef enum {

  CURLINFO\_TEXT = 0, /\* 0 信息 \*/

  CURLINFO\_HEADER\_IN, /\* 1 信息输入头\*/

  CURLINFO\_HEADER\_OUT, /\* 2 信息输出头\*/

  CURLINFO\_DATA\_IN, /\* 3 数据输入\*/

  CURLINFO\_DATA\_OUT, /\* 4 数据输出\*/

  CURLINFO\_SSL\_DATA\_IN, /\* 5 SSL加密数据(二进制)输入\*/

  CURLINFO\_SSL\_DATA\_OUT, /\* 6 SSL加密数据(二进制)输出\*/

  CURLINFO\_END

} curl\_infotype;

/\* 回调函数接口定义\*/

int debug\_callback(CURL \*handle,

  curl\_infotype type,

  char \*data,

  size\_t size,

  void \*userptr);

/\* curl调用回调函数进行debug日志输出 \*/

CURLcode curl\_easy\_setopt(CURL \*handle, CURLOPT\_DEBUGFUNCTION,

  debug\_callback);

static

void dump(const char \*text,

  FILE \*stream, unsigned char \*ptr, size\_t size)

{

  size\_t i;

  size\_t c;

  unsigned int width=0x10;

  fprintf(stream, "%s, %10.10ld bytes (0x%8.8lx)\n",

  text, (long)size, (long)size);

  for(i=0; i<size; i+= width) {

  fprintf(stream, "%4.4lx: ", (long)i);

  /\* show hex to the left \*/

  for(c = 0; c < width; c++) {

  if(i+c < size)

  fprintf(stream, "%02x ", ptr[i+c]);

  else

  fputs(" ", stream);

  }

  /\* show data on the right \*/

  for(c = 0; (c < width) && (i+c < size); c++) {

  char x = (ptr[i+c] >= 0x20 && ptr[i+c] < 0x80) ? ptr[i+c] : '.';

  fputc(x, stream);

  }

  fputc('\n', stream); /\* newline \*/

  }

}

static

int my\_trace(CURL \*handle, curl\_infotype type,

  char \*data, size\_t size,

  void \*userp)

{

  const char \*text;

  (void)handle; /\* prevent compiler warning \*/

  (void)userp;

  switch (type) {

  case CURLINFO\_TEXT:

  fprintf(stderr, "== Info: %s", data);

  default: /\* in case a new one is introduced to shock us \*/

  return 0;

  case CURLINFO\_HEADER\_OUT:

  text = "=> Send header";

  break;

  case CURLINFO\_DATA\_OUT:

  text = "=> Send data";

  break;

  case CURLINFO\_SSL\_DATA\_OUT:

  text = "=> Send SSL data";

  break;

  case CURLINFO\_HEADER\_IN:

  text = "<= Recv header";

  break;

  case CURLINFO\_DATA\_IN:

  text = "<= Recv data";

  break;

  case CURLINFO\_SSL\_DATA\_IN:

  text = "<= Recv SSL data";

  break;

  }

  dump(text, stderr, (unsigned char \*)data, size);

  return 0;

}

int main(void)

{

CURL \*curl;

CURLcode res;

struct data config;

config.trace\_ascii = 1; /\* enable ascii tracing \*/

curl = [curl\_easy\_init](https://curl.se/libcurl/c/curl_easy_init.html)();

if(curl) {

[curl\_easy\_setopt](https://curl.se/libcurl/c/curl_easy_setopt.html)(curl, [CURLOPT\_DEBUGFUNCTION](https://curl.se/libcurl/c/CURLOPT_DEBUGFUNCTION.html), my\_trace);

[curl\_easy\_setopt](https://curl.se/libcurl/c/curl_easy_setopt.html)(curl, [CURLOPT\_DEBUGDATA](https://curl.se/libcurl/c/CURLOPT_DEBUGDATA.html), &config);

/\* 输出详细信息DEBUGFUNCTION 才生效\*/

[curl\_easy\_setopt](https://curl.se/libcurl/c/curl_easy_setopt.html)(curl, [CURLOPT\_VERBOSE](https://curl.se/libcurl/c/CURLOPT_VERBOSE.html), 1L);

/\* example.com is redirected, so we tell libcurl to follow redirection \*/

[curl\_easy\_setopt](https://curl.se/libcurl/c/curl_easy_setopt.html)(curl, [CURLOPT\_FOLLOWLOCATION](https://curl.se/libcurl/c/CURLOPT_FOLLOWLOCATION.html), 1L);

[curl\_easy\_setopt](https://curl.se/libcurl/c/curl_easy_setopt.html)(curl, [CURLOPT\_URL](https://curl.se/libcurl/c/CURLOPT_URL.html), "https://example.com/");

res = [curl\_easy\_perform](https://curl.se/libcurl/c/curl_easy_perform.html)(curl);

/\* Check for errors \*/

if(res != CURLE\_OK)

fprintf(stderr, "[curl\_easy\_perform](https://curl.se/libcurl/c/curl_easy_perform.html)() failed: %s\n",

[curl\_easy\_strerror](https://curl.se/libcurl/c/curl_easy_strerror.html)(res));

/\* always cleanup \*/

[curl\_easy\_cleanup](https://curl.se/libcurl/c/curl_easy_cleanup.html)(curl);

}

return 0;

}

**2. curl\_easy\_getinfo**

curl = curl\_easy\_init();

  if(curl) {

  curl\_easy\_setopt(curl, CURLOPT\_URL, "https://www.example.com/");

  res = curl\_easy\_perform(curl);

  if( [CURLE\_OK](https://curl.se/libcurl/c/libcurl-errors.html#CURLEOK) == res) {

  char \*ct;

  /\* 询问内容类型 \*/

  res = curl\_easy\_getinfo(curl, CURLINFO\_CONTENT\_TYPE, &ct);

  if(( [CURLE\_OK](https://curl.se/libcurl/c/libcurl-errors.html#CURLEOK) == res) && ct)

  printf("我们收到 Content-Type: %s\n", ct);

  }

  /\* 总是清理 \*/

  curl\_easy\_cleanup(curl);

  }

备注：curl\_easy\_getinfo的第三个参数可以是long 指针，char\* , struct curl\_slist \* 或者double指针，根据需要获取的实际数据来填充。

==========================================================================================================================

**附件发送(图片\文件\视频)**

情景1: 发送纯附件(单附件：图片，文件，视频，压缩包)

情景2: 发送多附件

情景3: 发送附件+文本(纯文本&HTML)

**情景1: 发送纯附件(单附件：图片，文件，视频，压缩包)**

curl = curl\_easy\_init();

if (curl) {

// 初始化mime结构

mime = curl\_mime\_init(curl);

// 在mime结构中添加part

part = curl\_mime\_addpart(mime);

curl\_mime\_data(part, inline\_html, CURL\_ZERO\_TERMINATED);

curl\_mime\_type(part, "text/html");

curl\_easy\_setopt(curl, CURLOPT\_MIMEPOST, mime);

// 发送

res = curl\_easy\_perform(curl);

// 释放

curl\_easy\_cleanup(curl);

curl\_mime\_free(mime);

}

**情景2: 发送多附件(同情景一)**

**情景3: 发送附件+文本(纯文本&HTML)**

curl = curl\_easy\_init();

if (curl) {

// 构建头信息

curl\_easy\_setopt(curl, CURLOPT\_HTTPHEADER, headers);

// 初始化mime结构

mime = curl\_mime\_init(curl);

text = curl\_mime\_init(curl);

// 在text mime结构添HTML part

part = curl\_mime\_addpart(text);

curl\_mime\_data(part, inline\_html, CURL\_ZERO\_TERMINATED);

curl\_mime\_type(part, "text/html");

// 在mime结构中添加纯文本part

part = curl\_mime\_addpart(text);

curl\_mime\_data(part, inline\_text, CURL\_ZERO\_TERMINATED);

/\* Create the inline part. \*/

part = curl\_mime\_addpart(mime);

curl\_mime\_subparts(part, text);

curl\_mime\_type(part, "multipart/alternative");

slist = curl\_slist\_append(NULL, "Content-Disposition: inline");

curl\_mime\_headers(part, slist, 1);

// 在mime结构中添加part

part = curl\_mime\_addpart(mime);

curl\_mime\_data(part, inline\_html, CURL\_ZERO\_TERMINATED);

curl\_mime\_type(part, "text/html");

curl\_easy\_setopt(curl, CURLOPT\_MIMEPOST, mime);

// 发送

res = curl\_easy\_perform(curl);

// 释放

curl\_easy\_cleanup(curl);

curl\_mime\_free(mime);

}

==========================================================================================================================

**代理Proxy**

相关接口调用：

CURLOPT\_PRE\_PROXY 要使用的预代理主机

CURLOPT\_PROXY 使用代理

CURLOPT\_PROXYAUTH HTTP 代理身份验证方法

CURLOPT\_PROXYHEADER 要传递给代理的一组 HTTP 标头

CURLOPT\_PROXYPASSWORD 用于代理身份验证的密码

CURLOPT\_PROXYPORT 代理监听的端口号

CURLOPT\_PROXYTYPE 代理协议类型

CURLOPT\_PROXYUSERNAME 用于代理身份验证的用户名

CURLOPT\_PROXYUSERPWD 用于代理身份验证的用户名和密码

CURLOPT\_PROXY\_CAINFO 代理证书颁发机构 (CA) 捆绑包的路径

CURLOPT\_PROXY\_CAINFO\_BLOB PEM 格式的代理证书颁发机构 (CA) 捆绑包

CURLOPT\_PROXY\_CAPATH 持有 HTTPS 代理 CA 证书的目录

CURLOPT\_PROXY\_CRLFILE HTTPS代理证书吊销列表文件

CURLOPT\_PROXY\_ISSUERCERT 代理颁发者 SSL 证书文件名

CURLOPT\_PROXY\_ISSUERCERT\_BLOB 来自内存 blob 的代理颁发者 SSL 证书

CURLOPT\_PROXY\_KEYPASSWD 代理私钥的密码

CURLOPT\_PROXY\_PINNEDPUBLICKEY https 代理的固定公钥

CURLOPT\_PROXY\_SERVICE\_NAME 代理认证服务名称

CURLOPT\_PROXY\_SSLCERT HTTPS 代理客户端证书

CURLOPT\_PROXY\_SSLCERTTYPE 代理客户端 SSL 证书的类型

CURLOPT\_PROXY\_SSLCERT\_BLOB 来自内存 blob 的 SSL 代理客户端证书

CURLOPT\_PROXY\_SSLKEY HTTPS代理客户端证书的私钥文件

CURLOPT\_PROXY\_SSLKEYTYPE 代理私钥文件的类型

CURLOPT\_PROXY\_SSLKEY\_BLOB 来自内存 blob 的代理证书的私钥

CURLOPT\_PROXY\_SSLVERSION 首选 HTTPS 代理 TLS 版本

CURLOPT\_PROXY\_SSL\_CIPHER\_LIST 用于 HTTPS 代理的密码

CURLOPT\_PROXY\_SSL\_OPTIONS HTTPS 代理 SSL 行为选项

CURLOPT\_PROXY\_SSL\_VERIFYHOST 针对主机验证代理证书的名称

CURLOPT\_PROXY\_SSL\_VERIFYPEER 验证代理的 SSL 证书

CURLOPT\_PROXY\_TLS13\_CIPHERS 用于代理 TLS 1.3 的密码套件

CURLOPT\_PROXY\_TLSAUTH\_PASSWORD 用于代理 TLS 身份验证的密码

CURLOPT\_PROXY\_TLSAUTH\_TYPE HTTPS 代理 TLS 身份验证方法

CURLOPT\_PROXY\_TLSAUTH\_USERNAME 用于代理 TLS 身份验证的用户名

CURLOPT\_PROXY\_TRANSFER\_MODE 将 FTP 传输模式附加到代理的 URL

**代理流程：**

curl = curl\_easy\_init();

if(curl)

{

curl\_easy\_setopt(curl, CURLOPT\_URL, "smtp://mail.example.com");

// 设置代理

curl\_easy\_setopt(curl, CURLOPT\_PRE\_PROXY, "socks5://36.138.196.240:83");

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_FROM, FROM\_ADDR);

recipients = curl\_slist\_append(recipients, TO\_ADDR);

recipients = curl\_slist\_append(recipients, CC\_ADDR);

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_RCPT, recipients);

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, payload\_source);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &upload\_ctx);

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

/\* Send the message \*/

res = curl\_easy\_perform(curl);

/\* 释放资源 \*/

curl\_slist\_free\_all(recipients);

curl\_easy\_cleanup(curl);

}

**相关参考资料：**

代理相关网址：<https://www.proxydocker.com/zh/socks5-list/country/China>

测试TCP连接端口工具：<https://www.elifulkerson.com/projects/tcping.php>

设置邮箱代理安全性：<https://blog.csdn.net/kissradish/article/details/108470880>