SMTP Simple Mail Transport Protocol

Transfer von E-Mails, Erweiterungen von SMTP

Dirk Geschke

Linux User Group Erding

27. Juni 2007

Gliederung

Die Anfänge von E-Mail.
CTSS

- CTSS: Mehrere Benutzer auf einem System
  $\longrightarrow$ Kommunikation via Dateien
  1965: MAIL-Programm, Möglichkeit Texte an Dateien anderer Benutzer anzuhängen (Noel Morris und Tom Van Vleck)
- TENEX: 2 Computer via CPYNET verbunden
  $\longrightarrow$ Dateiaustausch zwischen 2 Systemen möglich
  $\longrightarrow$ SNDMSG und READMAIL von Ray Tomlinson
  $\longrightarrow$ @-Zeichen zur Trennung Benutzer - System

E-Mail Standard:

- Es entstanden viele unabhängige inkompatible Mailsysteme, siehe z.B. Appendix A von RFC-808
- Einheitlicher Standard für den Aufbau und Headerfelder wurde notwendig
  $\longrightarrow$ RFC-822: Standard for the Format of ARPA Internet Text Messages.
- Standard für das Übertragunsprotokoll spezifiziert RFC-821: Simple Mail Transfer Protocol.
- Aktuelle Versionen sind RFC-2821 und RFC-2822

Bestandteile einer E-Mail

Eine E-Mail besteht aus 3 Teilen:

- Envelope: gehört zum Transportprotokoll
- Header: Der Briefkopf einer E-Mail
- Body: Der eigentliche Inhalt der E-Mail
- Der Envelope ist normalerweise nicht Bestandteil des Headers bzw. der E-Mail, d.h. der Empfänger bekommt diesen nicht zu sehen.

Allgemeines

Allgemeines über das Protokoll:

- TCP-basiertes Protokoll, Port: 25
- Mail-Exchanger wird via MX-Record im DNS gesucht:

```
lug-erding.de. IN MX 10 mail.lug-erding.de.
```

- Niedrigster Prioritätswert bekommt den ersten Zustellversuch.
- Existiert kein MX-Record so wird der A-Record verwendet.
- Jeder SMTP-Server auf dem *Weg* einer E-Mail wird als hop bezeichnet
- Mailserver mit Protokollumsetzung nennt man gateway

Allgemeines

Allgemeines über das Protokoll:

- Mailserver trägt die Verantwortung für E-Mail, d.h. entweder wird eine E-Mail korrekt zugestellt oder es wird ein Fehler generiert.
- Server antwortet auf jede Eingabe des Clients mit einem OK, einer Aufforderung weitere Daten zu senden, einem temporären oder permanenten Fehler.
- Ausnahme: PIPELINING wird vom Server unterstützt
- Die Kommunikation ist in der Regel zeilenorientiert, Abschluß ist immer CRLF (#0D#0A).
- Server antwortet immer mit einem dreistelligen Zahlencode, einem Leerzeichen und einem ASCII-Text

Allgemeines

Allgemeines über das Protokoll:

- Zeilenlänge ist auf 1000 Zeichen *inklusive* CRLF begrenzt.
- Client-Kommandos sind *case-insensitive* und bestehen aus reinem ASCII.
- Empfehlung: Kommandos *groß* schreiben, einige Server verlangen dies entgegen RFC-821.
- SMTP erlaubt generell nur 7-Bit Daten, 8. Bit darf gelöscht oder die Annahme verweigert werden.
- Extended SMTP ermöglicht auch 8-Bit für den Mailbody
- 8-Bit Zeichen im Mailheader sind nur via MIME in 7-Bit Darstellung möglich.

E-Mail Rollen

Es gibt mehrere Rollen die bei E-Mail eine Rolle spielen

- **MSA**: **M**ail **S**ubmission **A**gent – Mailinitiator
- **MTA**: **M**ail **T**ransfer **A**gent – Mailserver zu Mailserver
- **MDA**: **M**ail **D**elivery **A**gent – ausliefernder Mailserver
- **MUA**: **M**ail **U**ser **A**gent – Mailclient
- **Gateway**: Vermittelt E-Mails zwischen 2 Transportsystemen, dabei ist ein Umschreiben der Inhalte möglich.
  $\longrightarrow$ Firewalls die Adressen umschreiben sind Gateways auch wenn sie auf beiden Seiten SMTP sprechen.

Die Unterscheidung wird aber eigentlich selten verwendet.

Kommunikation mit einem Mailserver

1. Client öffnet TCP-Verbindung zu Port 25 des Servers
2. Server antwortet mit Eröffnungsmitteilung, $\longrightarrow$ kann Software und Version zur Fehlersuche enthalten, z.B.

```
220 mail.lug-erding.de ESMTP \
Sendmail 8.13.8/8.13.8/Debian-2; \
Thu, 21 Jun 2007 09:20:30 +0200; \
(No UCE/UBE) logging access from: \
[91.64.98.12](FAIL)-[91.64.98.12]
```

3. Client sendet EHLO-Kommando mit FQDN des Clients, z.B.:

```
EHLO geschke.linuxhome.org
```

Falls der Server oder Client dies nicht unterstützt wird HELO verwendet (RFC-822)

Kommunikation mit einem Mailserver

4. Server antwortet mit unterstützen Kommandos / Erweiterungen (bei EHLO), z.B.:

```
250-mail.lug-erding.de Hello [92.64.98.12], plea
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-AUTH DIGEST-MD5 CRAM-MD5
250-DELIVERBY
250 HELP
```

Minus-Zeichen an der 4. Stelle: es gibt weitere Zeilen!

Kommunikation mit einem Mailserver

5. **MAIL FROM:** <*reverse-path*> [<*mail-parameter*>]
   Der *reverse-path* enthält idR den Absender, Fehler-E-Mails werden dorthin gesendet z.B.:

   ```
   MAIL FROM: <Dirk@geschke-online.de>
   ```

6. Wird der Sender akzeptiert so sendet der Server 250 OK
7. **RCPT TO:** <*forward-path*> [<*rcpt-parameter*>]
   Der *forward-path* enthält die Empfängeradresse, z.B.:

   ```
   RCPT TO: <Dirk@lug-erding.de>
   ```

8. Wird der Empfänger akzeptiert so sendet der Server 250 OK oder 550 no such user, z.B.:

   ```
   250 2.1.5 <Dirk@lug-erding.de>... Recipient ok
   ```

Kommunikation mit einem Mailserver

9. Client sendet **DATA**
10. Server antwortet idealerweise mit 354 go ahead, z.B:

```
354 Enter mail, end with "." on a line by itself
```

11. Client überträgt Daten zeilenweise, Mailheader und Mailbody sind durch Leerzeile getrennt. Den Abschluß bildet die Sequenz **“CRLF . CRLF”**, z.B.:

```
Subject: Just a Test
From: <Dirk@geschke-online.de>

nix besonderes
.
```

Kommunikation mit einem Mailserver

12. Server sendet 250 OK, z.B.:

```
250 2.0.0 l5L81DFD032735 Message accepted \
for delivery
```

13. Beenden der Übertragung, Client sendet QUIT
14. Server sendet 221 OK und schließt die Verbindung:

```
221 2.0.0 mail.lug-erding.de closing connection
```

Das ist im wesentlichen alles, es können aber auch Fehlercodes ausgegeben werden und es gibt noch Kommandos für die Fehlersuche sowie ein paar Erweiterungen.

Kommandos für Mailtransport

EHLO Extended Hello:

```
EHLO Domainname
```

Server antwortet dann mit

```
250-domain greeting
250-unterstuetzte ESMTP Kommandos
...
250 letztes Kommando
```

Wenn von älteren Clients nicht unterstützt muß

```
HELO Domainname
```

möglich sein $\longrightarrow$ kein ESTMP!

Kommandos für Mailtransport

MAIL Angabe des Senders:

`MAIL FROM: <reverse-path> [Parameter]`

Der reverse-path darf auch "<>" sein zur Vermeidung von *double-bounces*

RCPT Recipient, Empfänger

`RCPT TO: <forward-path> [Parameter]`

Eine Adresse muß immer existieren: postmaster, dies ist die einzige Adresse die auch ohne FQDN angenommen werden muß!
Forward-path bei Mailrouting:

`RCPT TO: <@host.de,@foo.de:joe@bar.org>`

$\longrightarrow$ obsolet, Server darf direkt zustellen.

Kommandos für Mailtransport

DATA Einleitung des Datentransfers

- normalerweise antwortet der Server mit 354
- Daten werden zeilenweise gesendet und mit CRLF abgeschlossen
- Maximallänge ist 1000 Zeichen inklusive CRLF
- Ende der Übertragung: CRLF.CRLF
- reguläre Punkte am Zeilenanfang werden verdoppelt.
- Mail akzeptiert: Server fügt TRACE-Eintrag (“**Received:**”) am **Anfang** in den Header ein

Kommandos für Mailtransport

RSET Reset, zurücksetzen der Verbindung

- gespeicherte Sender, Empfänger sowie Maildaten müssen verworfen werden
- Server antwortet mit 200 OK.
- Server muß die Verbindung bestehen lassen!

HELP Server muß dies ohne Argumente unterstützen, kann dies aber auch mit. In der Regel werden alle unterstützten Kommandos aufgelistet.

NOOP Server liefert nur 200 OK zurück

QUIT beendet die Verbindung

Kommandos für die Fehlersuche

VRFY **Verify**: Überprüfen von Mailadressen, z.B.:

```
VRFY geschke
250 2.1.5 Dirk Geschke\
  <geschke@majestix.physik.home>
```

EXPN **Expand**: Expandieren von Mailinglisten

```
VRFY liste
250 2.1.5 <liste@majestix.physik.home>
EXPN liste
250-2.1.5 Dirk Geschke <|/bin/procmail>
250-2.1.5 <dirk@lug-erding.de>
250 2.1.5 <linux@lug-erding.de>
```

Sicherheitseinstellung

Aus Sicherheitsgründen wird dies bei den meisten Mailservern deaktiviert, z.B.:

```
EXPN linux@lug-erding.de
502 5.7.0 Sorry, we do not allow this operation
VRFY linux@lug-erding.de
252 2.5.2 Cannot VRFY user; try RCPT to\
 attempt delivery (or try finger)
```

Trotzdem muß das Kommando **VRFY** implementiert sein!

Syntax der Argumente

Für reverse-path bzw. forward-path gilt:

Path	“<” [A-d-l “:”] Mailbox “>”
A-d-l	“@” domain *(“,” A-d-l)
Mailbox	local-part “@” domain
local-part	dot-string / quoted string
domain	(sub-domain 1*(“.” sub-domain)) / address-lit
address-lit	“[“ IPv4-address-literal | IPv6-address-literal | general-address-literal“]”

Reihenfolge der Kommandos

1. EHLO oder HELO
2. MAIL bzw. **SEND**, **SOML** oder **SAML** (obsolet)
3. RCPT, es können mehrere RCPT-Zeilen folgen
4. DATA
5. QUIT

- NOOP, HELP, EXPN, VRFY sowie RSET dürfen überall verwendet werden
  Empfehlung: zuerst EHLO verwenden
- eigene Kommandos dürfen verwendet werden, müssen aber mit X beginnen

Numerische Antwort-Codes des Servers

Die 1. Stelle besagt:

- 1yz Kommando akzeptiert aber noch nicht ausgeführt
- 2yz Kommando wurde erfolgreich ausgeführt
- 3yz positive Zwischenmeldung, weitere Daten erforderlich
- 4yz temporärer Fehler
- 5yz permanenter Fehler

Numerische Antwort-Codes des Servers

Die 2. Stelle besagt:

- x0z Syntax-Fehler: nicht implementiert, überflüssig, falsche Reihenfolge
- x1z Information: Antwort auf Informationsfrage wie Status oder Hilfe
- x2z Verbindung: Alles rund um die Verbindung
- x3z unspezifiziert
- x4z unspezifiziert
- x5z Mailsystem: Status des Empfangssystems bzgl. Mailtransfer oder anderer Mailaktionen

Die 3. Stelle gibt eine feinere Bedeutung pro Kategorie an.

Antwort-Codes in numerischer Reihenfolge

211 Systemstatus oder Hilfeantwort

214 Hilfemitteilung

220 *domain* Dienst bereit

221 *domain* beendet den Datenkanal

250 Mailaktion OK und vollständig

251 Empfänger nicht lokal, werde weiterleiten an *forward-path*

252 Der Benutzer kann nicht verifiziert werden, die Mail wird aber angenommen und es wird versucht diese weiterzuleiten

354 Warte auf Mail, Ende mit CRLF . CRLF

Antwort-Codes in numerischer Reihenfolge

421 *domain* Dienst nicht verfügbar, beende Verbindung (Timeout)

450 Mailbox nicht verfügbar

451 Bearbeitung abgebrochen, lokaler Fehler bei der Bearbeitung

452 Bearbeitung abgebrochen, ungenügender Systemspeicher

500 Syntaxfehler, Kommando nicht erkannt

501 Syntaxfehler bei Parametern

502 Kommando nicht implementiert

503 Falsche Reihenfolge der Kommandos

504 Kommandoparameter nicht implementiert

Antwort-Codes in numerischer Reihenfolge

550 Angeforderter Aktion nicht ausgeführt, Mailbox nicht verfügbar (z.B. existiert nicht, nicht zugreifbar oder Policy-Gründe)

551 Empfänger nicht lokal; bitte versuche *forward-path*

552 Bearbeitung abgebrochen, ungenügend Speicherplatz

553 Bearbeitung abgebrochen, Mailbox-Name nicht erlaubt oder mehrdeutig

554 Transaktion fehlgeschlagen, bei Eröffnung der Verbindug: Kein SMTP-Dienst verfügbar

Received-Zeile

Ein Server muß eine Received-Zeile hinzufügen

- FROM-Feld muß existieren und soll den Namen des Absendeservers aus EHLO Zeile enthalten sowie die IP-Adresse aus der TCP-Verbindung
- ID-Feld kann “@” enthalten (RFC-822), muß aber nicht
- FOR-Feld kann eine Liste von Empfängern (bei mehreren RCPT-Kommandos) enthalten, aus Sicherheitsgründen nicht wünschenswert (z.B. **BCC**)
- Server darf keine Received-Zeilen entfernen oder verändern (Firewall!)
- Neue Received-Zeile muß im Header vorangestellt werden
- Die Reihenfolge darf nicht geändert werden!

Received-Zeile

- Zeiten sollen mit Offset statt Zeitzone angegeben werden
- Es sollte die lokale Uhrzeit verwendet werden, das erleichtert die Fehlersuche
- Letzter SMTP-Server fügt return-path hinzu, diese enthält den Absender aus dem Envelope! (optional)
  $\longrightarrow$ Mailinglistenbetreiber für Fehlermails

Received-Zeile - Ein Beispiel

Beispiel für Received-Zeile:

```
Received: from mail.lug-erding.de
        (mail.lug-erding.de [89.110.147.240])
        by majestix.physik.home
        (8.13.4/8.13.4/Debian-3sarge3) with
        ESMTP id l5LFf6oZ027286
        for <dirk@geschke-online.de>;
        Thu, 21 Jun 2007 17:41:06 +0200
```

Mailserver als Relay

- MX-Records machen *Source-Routing* überflüssig
  $\longrightarrow$ Clients sollten kein Source-Routing verwenden
  $\longrightarrow$ Server dürfen dies ablehnen (Code 550), sie dürfen auch direkt zustellen.
- Wird E-Mail angenommen und kann nicht weitergeleitet werden so muß eine Fehlermail an den Absender aus dem Envelope gesendet werden.
- keine Fehlermail auf eine Fehlermail
- Fehlermails haben als Absendeadresse: <>
- Server braucht Mailheader nicht zu untersuchen oder verändern, lediglich eine Received-Zeile hinzufügen und Mail–Loops erkennen.

Mailserver als Gateway

- Umsetzung von Protokollen, z.B. SMTP $\longrightarrow$ X.400
- eventuell müssen Mailheader umgeschrieben werden
- Gateway muß Received-Zeile hinzufügen, darf aber keine ändern
- in *via*-Klausel Umgebungen und Protokolle vermerken
- Gateway muß bei Versand nach SMTP die Header dem Standard anpassen, insbesondere To, From, Cc, etc.
- Umwandlungen in nicht-SMTP-Systeme sollen sicherstellen, daß Fehlermails an die Envelope-Adresse gesendet werden und nicht an die From-Adresse des Mailheaders
- umgekehrt gilt das Gleiche!

Mailinglisten und Aliase

- SMTP-Server sollten beides unterstützen
- bei Listen: MAIL FROM: sollte den Listenverwalter angeben (wegen Fehlermails)

  Alias Mailserver ersetzt die Zieladresse im Envelope, der Rest bleibt unverändert

  Liste Zieladresse im Envelope werden durch die Adressen der Liste ersetzt, die Absendeadresse im Envelope wird die des Listenverwalters

8bit-MIMEtransport

Methode auch binäre Zeichen zu übertragen

- RFC-1652
- **EHLO**-Keyword 8BITMIME
- **MAIL**-Parameter: BODY
- Wert: **7BIT** / **8BITMIME**
- 8-Bit Werte im Mailbody erlaubt!
- Beschränkung auf 1000 Zeichen pro Zeile bleibt!

Checkpoint / Restart

Methode um unterbrochene Übertragungen wieder fortzusetzen, experimentell

- RFC-1845
- **EHLO**-Keyword CHECKPOINT
- **MAIL**-Parameter: TRANSID
- Wert: *transid-local* “@” *transid-domain*
- Server antwortet mit Bytes der bereits erhaltenen Daten
- wenig Aussicht auf Implementation -> DoS-Problem

Message Size Declaration

Möglichkeit die Maillänge im Vorfeld anzugeben

- RFC-1870
- **EHLO**-Keyword SIZE, optionaler Parameter: maximale Größe einer E-Mail
- **MAIL**-Parameter: SIZE= *Größe in Bytes*
- Server darf sich nicht auf diese Angabe verlassen!
- Vermeidung von Mailtransfer wenn Quota überschritten würden

Remote Message Queue Starting

Alternative zu (obsoletem) **TURN** um Mailtransfer anzustoßen

- RFC-1985
- **EHLO**-Keyword ETRN
- Kommando: ETRN *FQDN*
- Server startet daraufhin den Queue-Run für *FQDN*
- interessant für Server die nur gelegentlich *online* sind

Returning Enhanced Error Codes

Ergänzung zum dreistelligen Return-Code, granularer

- RFC-2034
- **EHLO**-Keyword ENHANCEDSTATUSCODES
- Außer der *Greeting*-Meldung erhalten alle Antworten einen weiteren Antwort-Code
- dieser ist granularer und wird für DSN verwendet

Authentication

Authentisierung von Clients

- RFC-2554
- **EHLO**-Keyword AUTH
- Parameter: Liste der unterstützten **SASL**-Methoden
- Kommando: AUTH *Auth-Mechanismus* [*initial-response*]
- **MAIL**-Parameter: AUTH=*auth-Absendeadresse*
- der optionale Parameter dient der Mitteilung an *trusted* Mailservern

Deliver By

Anweisung an den Server bis wann eine E-Mail ausgeliefert werden soll

- RFC-2852
- **EHLO**-Keyword DELIVERBY
- optionaler Parameter: minimale Zeit in Sekunden bei Option **R**
- **Mail**-Parameter: BY *Zeit* ; R / N / T
- Option **R**: failed DSN *delivery time expired*
- Option **N**: delayed DSN, wird trotzdem zugestellt
- Option **T**: Trace, Funktion: ?

Command Pipelining

Senden mehrerer Kommandos ohne jedesmal auf die Antwort zu warten

- RFC-2920
- **EHLO**-Keyword PIPELINING
- Antworten des Servers erfolgen am Ende Zeilenweise
- Idee: schnelleres flushen von Buffern
- besseres Verhalten bei fehlerhaften Servern
- müßte eigentlich vom Kernel bereits auf TCP-Ebene realisiert sein

Transmission of Large and Binary MIME messages

Möglichkeit die Zeilenbeschränkung beim Transfer zu umgehen und binäre Daten in Blöcken zu übertragen

- RFC-3030
- **EHLO**-Keyword CHUNKING
- Kommando: BDAT *chunk-size* [LAST]
- Blockgröße definiert das Ende der Daten
- keine Zeilenorientierung mehr notwendig
- Parameter **LAST** besagt: letzter Block
- **EHLO**-Keyword BINARYMIME
- **MAIL**-Paramter: BODY=BINARYMIME

Secure SMTP over TLS

Dient der verschlüsselten Übertragung zwischen 2 Mailservern

- RFC-3207
- **EHLO**-Keyword STARTTLS
- Kommando: STARTTLS
- direkt nach **STARTTLS** muß TLS ausgehandelt werden
- öffentlicher Server auf Port 25: Verwendung optional!
- Client/Server müssen entscheiden ob nicht-authentisierte Übertragung erlaubt ist
- nach erfolgreichem **TLS** ist die Verbindung zurückgesetzt $\longrightarrow$ Optionen können wegen Authentisierung anders sein
- Problem: keine End-to-End Verschlüsselung!

Delivery Status Notification

Standartisiertes Verfahren der Benachrichtigung, z.B. bei Fehlern

- RFC-3461
- **EHLO**-Keyword DSN
- **RCPT**-Parameter: NOTIFY = NEVER / SUCCESS / FAILURE / DELAY
- **RCPT**-Parameter: ORCPT = rfc822;*original-Empfänger*
- **MAIL**-Parameter: RET = FULL / HDRS
- **MAIL**-Parameter: ENVID = *envelope-ID*
- **DELAY** sendet **DSN** bei Verzögerungen, z.B. 4h E-Mails
- *envelope-ID* wird in einer **DSN** mitgesendet

Message Tracking

temporäre Speicherung von Transferdaten

- RFC-3885
- **EHLO**-Keyword MTRK
- **MAIL**-Parameter: MRTK = *Base64-ID* [: *Timeout*]
- *Timeout*: Sekunden die Trackinginformationen gespeichert werden sollen
- default: 8-10 Tage, Verwendung von **ENVID**

Content Conversion

bezieht sich auch auf den MIME-Inhalt, hier nur SMTP-Teil

- RFC-4141
- **EHLO**-Keyword CONPERM
- **MAIL**-Parameter: CONPERM
- $\longrightarrow$ Umwandlung der MIME-Daten erlaubt (*conversion permitted*)
- **EHLO**-Keyword CONNEG
- **RCPT**-Parameter: CONNEG
- $\longrightarrow$ Server antwortet mit **250-CONNEG** *Capability*
- $\longrightarrow$ *Capability* sind in RFC-2533 definiert

ENDE - Teil 1

Es fehlt noch

- Aufbau von Mailheader und Mailbody (RFC-2822)
- MIME (RFC-2045 bis RFC-2049)
- DSN-Format
- Spam, Spambekämpfung