Postfix: Status Quo current development an overview

Ralf Hildebrandt & Patrick Koetter

T-System Business Services & State Of Mind

LinuxForum 2006
Copenhagen, 04. March 2006

About T-System Business Services

- I just work there, I don't sell stuff
- I reckon we do sailing stuff
- strategic platform: Microsoft
- `http://www.t-systems.de`

About state of mind

- I own the place
- we do digital communication
- `http://www.state-of-mind.de`

the way development works

- stable version, changes every year or the other
- development happens in the snapshot versions
- fear not: snapshots are stable enough for everyday use
- if they’re good enough for me, they’re good enough for you :)

so whats up, Doc?

enhanced status codes according to RFC 3463

This allows answers in `access(5)` like:
`REJECT 5.7.1 You can't go here from there`,
which (hopefully) can be translated by the user's MUA (Outlook, Mozilla, etc.) into human-understandble format.

DSN support according to RFC 3461-3464

The sender of a message can specify how and if he/she wants to be notified about an (un)successful delivery.
MUAs support that!

limit on the number of TLS sessions per timeperiod

TLS-handshakes are computationally intensive.
The new limit
`smtpd_client_new_tls_session_rate_limit`
(disabled by default) reduces the implications of a DoS attack.

improved, but still not customizable logging

`delay=n` transformed into `delays=a/b/c/d`, where:

- a: the time before the queue manager (`qmgr`, including the time it took to transfer the mail
- b: the time inside the queue manager
- c: the time it took to establish the connection, incl. DNS, HELO and TLS-handshake
- d: the time it took to transfer the mail

All times have sub-second precision.
`conn_use=n` shows how often a connection has been re-used.
`relay=` now also shows, which port has been connected to on a host

log example

from my machine:

```
Feb  1 15:13:49 mail postfix/smtp[14123]: 94A06221600:
to=<recipient@charite.de>, relay=127.0.0.1[127.0.0.1]:10025,
conn_use=2, delay=3.4, delays=2.8/0.05/0/0.56, dsn=2.6.0,
status=sent (250 2.6.0 Ok, id=15344-04-2, from
MTA([127.0.0.1]:10026): 250 2.0.0 Ok: queued as AEC2222155B)
```

configurable bounce templates

Everybody complained about the lack of foreign-language (read: non-english) bounce messages within Postfix:
Now you can write your own bounce-messages
Achtung baby – german HOWTO:
`http://postfix.state-of-mind.de/bounce-templates`

sender-dependent relayhost settings

A feature for the home-user:
The relayhost is chosen in dependency of the envelope sender-address.
Postfix will use the proper credentials, too (user and password for SMTP-AUTH:

```
sender_dependent_relayhost_maps = hash:/etc/postf
smtp_sender_dependent_authentication = yes
```

shared code-base for LTMP and SMTP clients

The `smtp`-client now also implements the LMTP-protocol. This shared code-base introduces many new `lmtp_*`-parameters, which already existed as `smtp_*`-parameters.

- less redundancies
- better code-reuse
- consolidation of the system as a whole

new authentication plugin support

This is part of a greater “plugin-plan”.
Support for different authentication frameworks in the combined `smtp/lmtp`-client and `smtpd`-server

- Cyrus-SASL
- Dovecot-SASL
- (maybe!) Peter Bieringer’s “simple-auth”

Cyrus-SASL is generally considered problematic, since nobody understands it.
We don’t understand dovecot-authentication either, no questions please!

No MoRe CaptiLiZaTiOn

Postfix keeps the case when rewriting addresss using `canonical`, `virtual`, `relocated` and `generic` maps – even when using `$number`-style replacement operations in maps with regular expressions (`pcre` and `regexp`). `local(8)` and `virtual(8)` still downcase the key!

Refined TLS stuff

- Lutz's patch had some errors. Nobody noticed. It's amazing what you can get away with :)
- a new per-site TLS policy-mechanism works better against DNS-spoofing attacks (I don't want to go into details here!).

strict checks for privileged/non-privileged daemons

daemons that don't need root-privileges are not allowed to run with those privileges – they **must** be configured (in `master.cf`) to run unprivileged.

Antispam measures using policy delegation

Non-antique versions of Postfix may delegate the decision whether to accept or reject a message during a SMTP session to an external program.
Postfix hands over a set of META-data to the external program; the META-data consist of information gathered before the `DATA` stage.

During the SMTP session Postfix must decide what to do with the email:

- reject permanently
- reject with temporary error
- accept – and maybe bounce later
- delegate decision to external program (policy server) and execute decision

smtpd_*_restrictions

This example configures Postfix **without policy delegation**. SMTP session restrictions are handled using `smtpd_*_restrictions`:

```
smtpd_recipient_restrictions =
   permit_mynetworks
   permit_sasl_authenticated
   check_client_access hash:/etc/postfix/client_acc
   reject_unauth_destination
```

policy delegation example: Greylisting

RFC-compliant MTAs re-queue mail when they experience temporary errors on the receiving server side and will retry later.
Concept: Reject mail first with a temporary error!
“Real” MTAs will attempt a (second) delivery. Current Spambots have no mail queue - they drop the delivery attempt (and never come back).

input from the mail system

Input from the mail system:

- Client's IP address
- Envelope Sender Address
- Envelope Recipient Address

results

This results in the following policy:
If the policy server does not know the triplet made of the client's IP address, envelope sender address and envelope recipient address, Postfix's `smtpd` should be rejected with a temporary error code.
Postfix cannot do this on its own, but policy delegation can implement a policy service that does the trick.

Since version 2.1 Postfix `smtpd` daemon may delegate decision to external applications.
Why use an external process?

reasons

- complexity of implementation and configuration
  - `Mail::SPF::Query` makes up 64KB Perl-Code
  - policies may be long and complex
- security
  - no dependency on external libraries
  - strict security boundary between Postfix and “decision maker”
- high flexibility
  - free to choose any scripting language
    Perl, Python, . . .
  - decision can be delegated to a remote server
- performance may still be well

policyd protocol

The policyd protocol is rather simple:
The client send a query consisting of `name=value` pairs, separated by new lines and terminated by an empty line.
The server answers with a `name=value` pair, terminated by an empty line.

policyd protocol, client side

Postfix (client):

```
request=smtpd_access_policy
protocol_state=RCPT
protocol_name=SMTP
helo_name=some.domain.tld
queue_id=8045F2AB23
sender=foo@bar.tld
recipient=bar@foo.tld
client_address=1.2.3.4
client_name=another.domain.tld
instance=123.456.7
sasl_method=plain
sasl_username=you
sasl_sender=
```

policyd protocol, server side

Server (policy server): `action=defer_if_permit Greylisted`

$$emptyline$$

The policy server may respond any action defined in `smtpd access(5)` maps.

policyd protocol, possible responses

- `OK`
- `Number` (like OK)
- `4xx` Text (temporary error)
- `5xx` Text (permanent error)
- `REJECT` optional Text (error)
- `DEFER_IF_REJECT` optional Text
- `DEFER_IF_PERMIT` optional Text
- `restriction` (execute this restriction or smtpd_restriction_class)
- `DISCARD` optional Text
- `DUNNO`
- `FILTER` transport:nexthop
- `HOLD` optional Text
- `PREPEND` Headername: Headerwert

policyd protocol, possible responses

- policy servers may not tell Postfix about problems; they must log a warning instead and terminate connection.
- Postfix will then reply a generic, temporary error to the client. Upon retry by the client Postfix will contact the policy server again.
- policy server may run together with Postfix on the same machine. Postfix may use either TCP- or Unix-domain-sockets to communicate with the policy server.

policyd protocol, examples

Examples:

```
check_policy_service inet:127.0.0.1:9998
check_policy_service unix:/some/where/policy
check_policy_service unix:private/policy
```

policyd protocol, examples 2

Delegation to a policy service is configured adding the service in `smtpd_*_restrictions`. The following example configures a policy service at the end:

```
smtpd_recipient_restrictions =
permit_mynetworks
permit_sasl_authenticated
check_client_access
hash:/etc/postfix/client_access
reject_unauth_destination
check_policy_service inet:127.0.0.1:10023
```

Ready to use implementations

Michael Tokarev wrote a template for a policy server `http://www.corpit.ru/mjt/smtpdpolicy-template.pl` in PERL.

There are other, specialized policy servers focussing on specific tasks:

Ready to use implementations 2

- `gld` by Salim Gasmi
  Greylisting
  `http://www.gasmi.net/gld.html`
- `policy weight` by Norbert Felber
  Kind of a SpamAssassin replacement that weights RBLs, RHSBLs and HELO, but doesn't use an emails content to decide what to do.
  `http://www.policyd-weight.org`
- `Policyd` by Cami
  Policyd is an anti-spam plugin for Postfix (written in C) that does Greylisting, Sender-(envelope, SASL or host / ip)-based throttling (on messages and/or volume per defined time unit), Recipient rate limiting, Spamtrap monitoring / blacklisting, HELO auto blacklisting and HELO