Postfix Configuration and Administration

state-of-mind

LISA'07
Dallas, November 2007

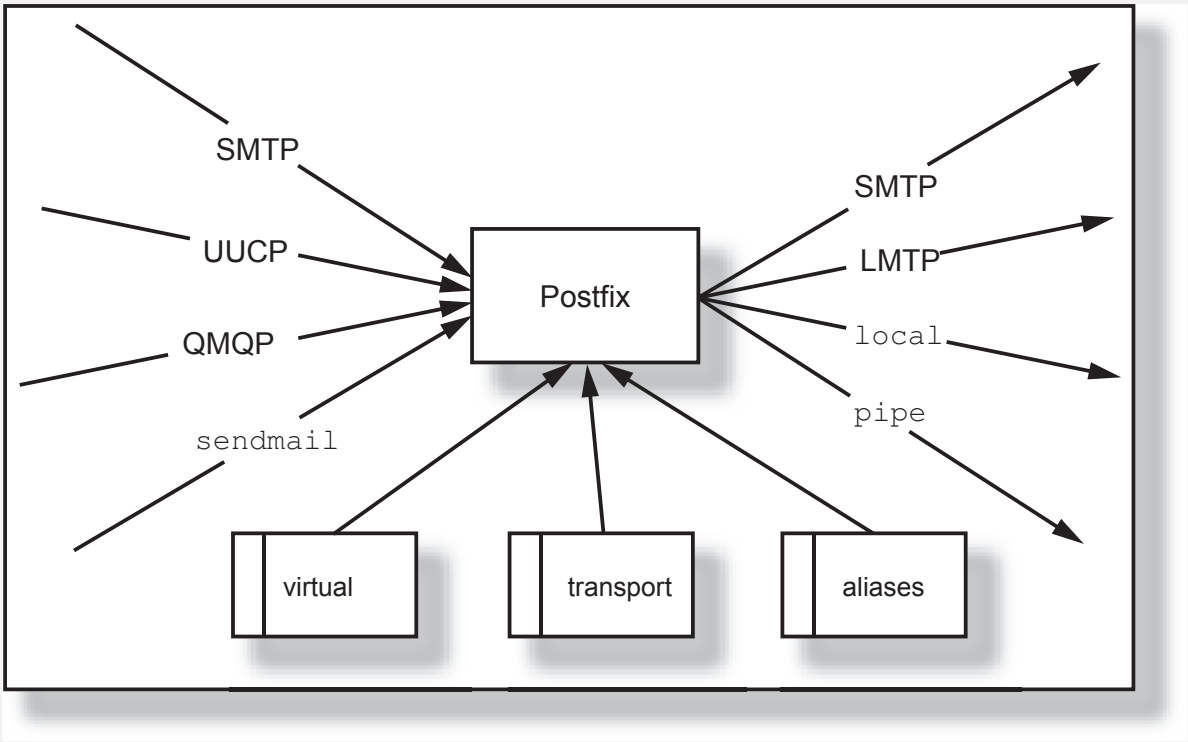


Figure: The Postfix Router

Postfix is a router

- It receives messages (packets) from a sender (source) and transports them closer to the recipient (target).
- Various interfaces are there to handle different protocols.
- Maps (routing tables) aid to select the appropriate interface and protocol.

Postfix is a firewall

- Check in- and outgoing traffic for basic requirements
- Enforce restrictions upon messages that do (not) match special criteria

- Postfix has a modular architecture
- Each daemon is specialized on one or only a few tasks
- Each daemon is run with the least priviledge required

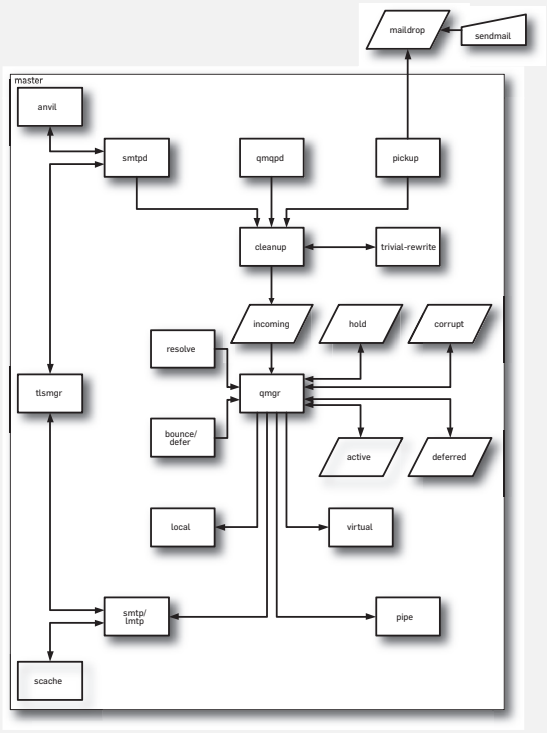


Figure: Postfix Daemons

Most important daemons

- `master`
  The master daemon is the brain of the Postfix mail system. It spawns all other daemons.
- `smtpd`
  The smtpd daemon (server) handles incoming connections.
- `smtp`
  The smtp client handles outgoing connections.
- `qmgr`
  The qmgr-Daemon is the heart of the Postfix mail system. It processes and controls all messages in the mail queues.
- `local`
  The local program is Postfix' own local delivery agent. It stores messages in mailboxes.

Maps help Postfix sort things out

- Accept or reject message?
- Who are my recipients?
- Which interface (read: transport) should I use to send this message?
- Is the sender permitted to relay?
- ...

For envelope sender and envelope recipient addresses

- `aliases`
- `virtual`
- `generic`
- `canonical`
- `relocated`

For dedicated transport settings

- `transports`

For SMTP communication control

- `access`

For content control

- `header_checks`
- `body_checks`
- `mime_header_checks`
- `nested_header_checks` (obscure)

Postfix can handle different map types

- Linear Maps
- Indexed Maps
- Dynamic Maps
- Network Maps


```
localpart+extension@subdomain.domain.tld
localpart@subdomain.domain.tld
subdomain.domain.tld
domain.tld
tld
localpart+extension@
localpart@
```

no match found

```
localpart+extension@subdomain.domain.tld
localpart@subdomain.domain.tld
localpart+extension
localpart
@subdomain.domain.tld
```

no match found


```
localpart+extension@subdomain.domain.tld
localpart@subdomain.domain.tld
subdomain.domain.tld
domain.tld
tld
*
```

no match found

Two configuration files configure Postfix runtime behavior:

- `main.cf`
  holds global configuration options. They will be applied to all instances of a daemon, unless they are overridden in `master.cf`
- `master.cf`
  defines runtime environment for daemons attached to services. Runtime behavior defined in `main.cf` may be overridden by setting service specific options.

SMTP requires a well configured environment. Postfix does not provide the environment.
Postfix expects the hosts OS and its services to provide the environment. A well configured host lays the ground for a well functioning Postfix!

- Hostname
- proper time
- DNS resolution
- DNS entries

What does Postfix need to provide basic services?

Configuring the basics addresses the following questions:

- Who am I?
- What's my name?
- Where am I?
- Whom am I responsible for?
- What should I append, if someone wants to send without a domainpart?
- Which interfaces should I listen on?
- Whom should I serve?

Commands you will use in everyday work with Postfix:

- `postalias`
- `postmap`
- `postconf`
- `postqueue`
- `postsuper`

- Describe your goal
- If possible tell how you want to achieve it
- Give current configuration using `postconf -n` output
- Give log excerpts that show your problem
- Tell what you have tried so far

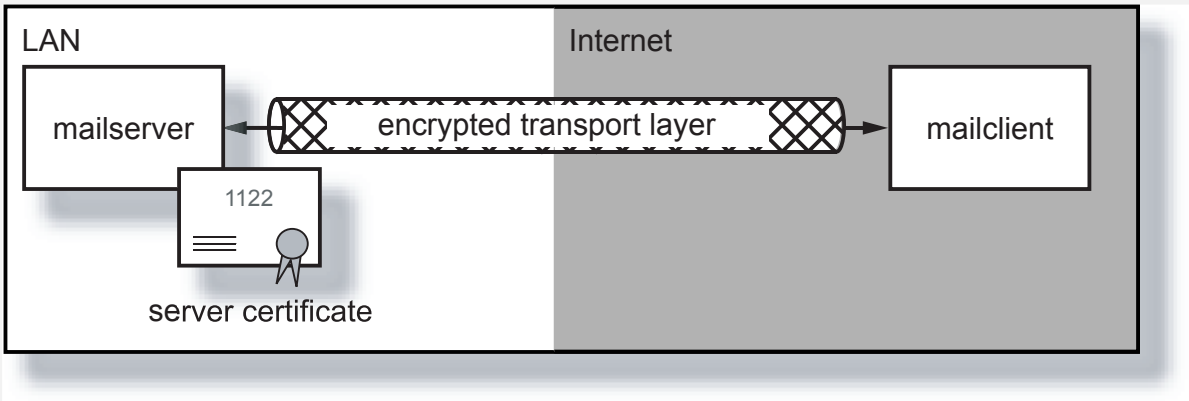


Figure: TLS in SMTP communication

Why use TLS anway?

- Privacy
- Integrity
- Authenticity
- Controlled Access

Common misconceptions:

- TLS only protects the communication between two hosts
- TLS only protects the transport, but not the storage


```
smtpd_tls_security_level = may
smtpd_tls_loglevel = 0
smtpd_tls_received_header = yes
smtpd_tls_key_file = /etc/postfix/smtp.key
smtpd_tls_cert_file = /etc/postfix/smtp.pem
smtpd_tls_CApath = /etc/pki/cacerts/
smtpd_tls_dh1024_param_file = /etc/postfix/dh_1024.pem
smtpd_tls_dh512_param_file = /etc/postfix/dh_512.pem
smtpd_tls_session_cache_database =
  btree:/var/spool/postfix/smtpd_scache
smtpd_tls_session_cache_timeout = 10800s
```

```
smtp_tls_security_level = may
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy
smtp_tls_loglevel = 0
smtp_tls_key_file = /etc/postfix/smtp.key
smtp_tls_cert_file = /etc/postfix/smtp.pem
smtp_tls_CApath = /etc/pki/cacerts/
smtp_tls_session_cache_database =
   btree:/var/spool/postfix/smtp_scache
smtp_tls_session_cache_timeout = 10800s
```


Relay control based on static IP-addresses is easy. But how would you deal with dynamic IP-addresses?

- VPN
  Seems like a little overkill for one service.
- `SMTP-after-(POP|IMAP) / (POP|IMAP)-before-SMTP`
  Uses another service to solve the problem and complicates the system.
- `TLS client certificates`
  Are a dream, but there's not enough clients to support it.
- `SMTP AUTH`
  Solves the problem where it arises.

Postfix does not process SMTP AUTH itself. Instead it either relies on the Cyrus SASL authentication framework or on the dovecot authentication service.
If Postfix uses Cyrus SASL, it can:

- offer SMTP AUTH (server-side, `smtpd`)
- use SMTP AUTH (client-side, `smtp`)
- control usage of the envelope sender

Postfix Configuration and Administration
Relay control
SMTP AUTH

The dovecot authentication implementation provides only server-side functionality. Using dovecot Postfix can:

- offer SMTP AUTH (server-side, `smtpd`)
- control usage of the envelope sender

Client certificates are **not** sent by default.

```
smtpd_tls_ask_ccert = yes
```

Three ways to permit relaying based upon client-certificate are available:

- `permit_tls_clientcerts`
- `permit_tls_all_clientcerts`
- `check_ccert_access type:table`


```
% openssl x509 -noout -fingerprint -md5 -in \
  client_certificates/1_cert.pem
MD5 Fingerprint=44:22:00:38:76:87:87:F6:67:27:5C:FB:D8:A5:75:9A

smtpd_recipient_restrictions =
   ...
   permit_mynetworks
   check_ccert_access hash:/etc/postfix/relay_certificates
   reject_unauth_destination
   ...
```

Every delivery attempt tries to answer the question:

To which host:user should I deliver localpart@domainpart messages?

local domain	localpart	domainpart	user	host
virtual alias domain	localpart	domainpart	user	host
virtual mailbox domain	localpart	domainpart	user (virtual)	host
relay domain	localpart	domainpart	user	host

Postfix Configuration and Administration
Multi-domain configurations
What's in a namespace?

The four results in the following namespace variations within Postfix:

- `local domain`
  A local domain has a fixed domainpart and host. Localparts are dynamic and delivery tries to match a system user.
- `virtual alias domain`
  A virtual alias domain has a fixed host. Localparts and domainparts are dynamic and delivery tries to match a system user.

- `virtual mailbox domain`
  A virtual mailbox domain has a fixed host. Localparts and domainparts are dynamic and delivery tries to match a virtual user.
- `relay domain`
  In a relay domain everything is dynamic. At least domainparts and hosts are known and will be sent to a remote host

A local domain name maps a domain name to local system users.

Virtual alias domains map additional domain names to local system users.

- easily done
- number of system users is limited (at least on Linux)

Virtual users are in no relation to system users except for the (read: usually one) UID and GID required to write messages to and read messages from a virtual user's mailbox.
Virtual mailbox domains use the Postfix virtual daemon for local delivery. It requires special configuration, since virtual has no access to $ENV:

- Where are mails stored?
- What's the recipients mailbox?
- Which mailbox format should be used?
- Which UID should be used to access the recpients mailbox?
- Which GID should be used to access the recpients mailbox?

A simple relay host configuration answers two questions:

- Do I need to accept mail for this domain?
- What's the next hop where I should transport the message to?

```
relay_domains = hash:/etc/postfix/relay_domains
```


per-domain transport

transport tables are evaluated before any other table!

```
mail.example.com :[gateway.example.com]
example.com      smtp:bar.example:2025
.example.com     error:mail for *.example.com is not deliverable
```

dynamic per-recipient transport

```
# LDAP connection parameters
version = 3
bind = yes
bind_dn = uid=postfix,ou=services,dc=example,dc=com
bind_pw = secret

# LDAP search and result configuration
server_host =
        ldap://ldap1.example.com
        ldap://ldap2.example.com

search_base = ou=people,dc=example,dc=com
query_filter = (|(&(mail=%s)(proxy=TRUE))(maildrop=%s))
result_format = mailstore:[%s]
result_attribute = proxyHostname
```


- What happens if you add more than one domain in mydestination?
- Why use virtual tables for local domains?

Postfix offers several methods to control message flow:

	internal methods	external methods
SMTP communication	smtpd_helo_restrictions smtpd_client_restrictions smtpd_sender_restrictions smtpd_recipient_restrictions smtpd_data_restrictions smtpd_end_of_data_restrictions smtpd_etrn_restrictions	check_policy
message content	header_checks body_checks mime_header_checks nested_header_checks	content_filter smtpd_proxy_filter
SMTP communication & message content	– none –	smtpd_milters non_smtpd_milters

- Postfix provides a trigger for each SMTP communication stage
- The trigger may evaluate one or more restrictions

In theory one would evaluate and act upon a restriction at the corresponding SMTP stage, but in practice the earliest moment to evaluate is after the first recipient has been submitted.

Moment of evaluation

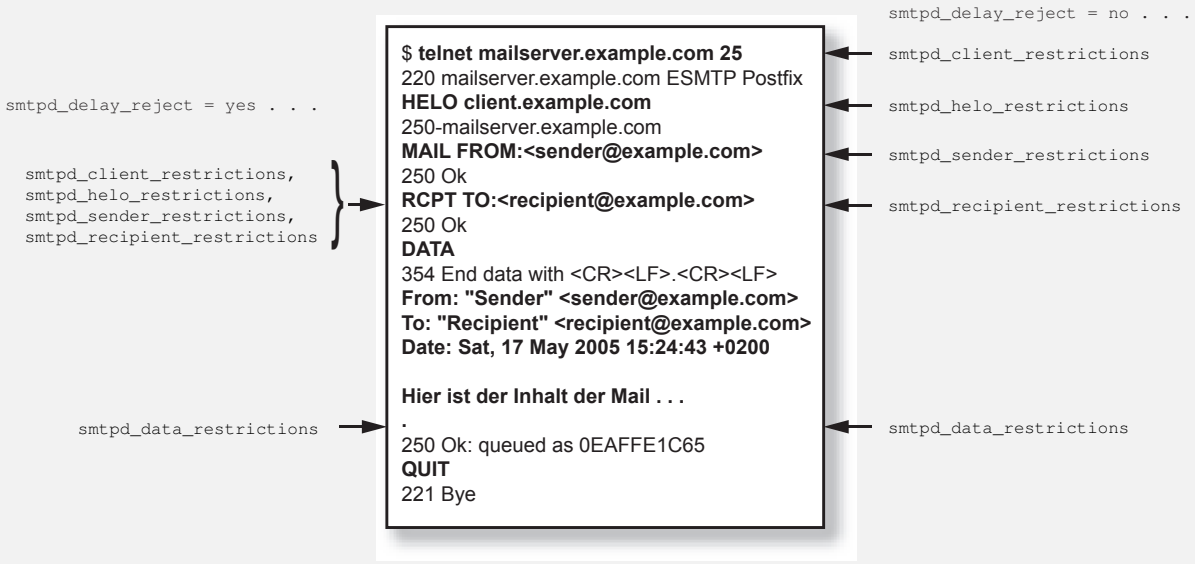
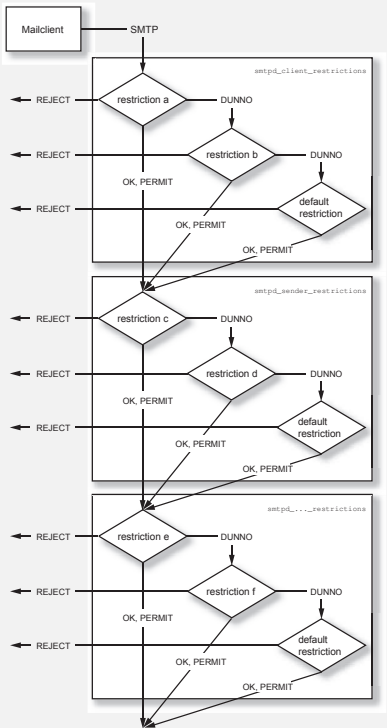


Figure: Moment of evaluation

Order of processing

The order in which single restrictions are listed is important:



The Postfix `smtpd` daemon delegates the decision what to do with the message to an external service:

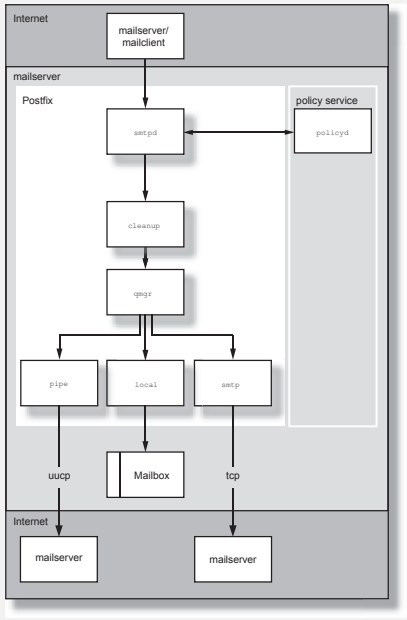


Figure: Policy Service

A simple protocol feeds an external service with SMTP communication meta data. Here is an example of all the attributes that the Postfix SMTP server sends in a delegated SMTPD access policy request:
Postfix version 2.1 and later:

```
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
recipient_count=0
client_address=1.2.3.4
client_name=another.domain.tld
reverse_client_name=another.domain.tld
instance=123.456.7
```

Postfix version 2.2 and later:

```
sasl_method=plain
sasl_username=you
sasl_sender=
size=12345
ccert_subject=solaris9.porcupine.org
ccert_issuer=Wietse+20Venema
ccert_fingerprint=C2:9D:F4:87:71:73:73:D9:18:E7:C2:F3:C1:DA:6E:04
```


Postfix version 2.3 and later:

```
encryption_protocol=TLSv1/SSLv3
encryption_cipher=DHE-RSA-AES256-SHA
encryption_keysize=256
etrn_domain=
[empty line]
```

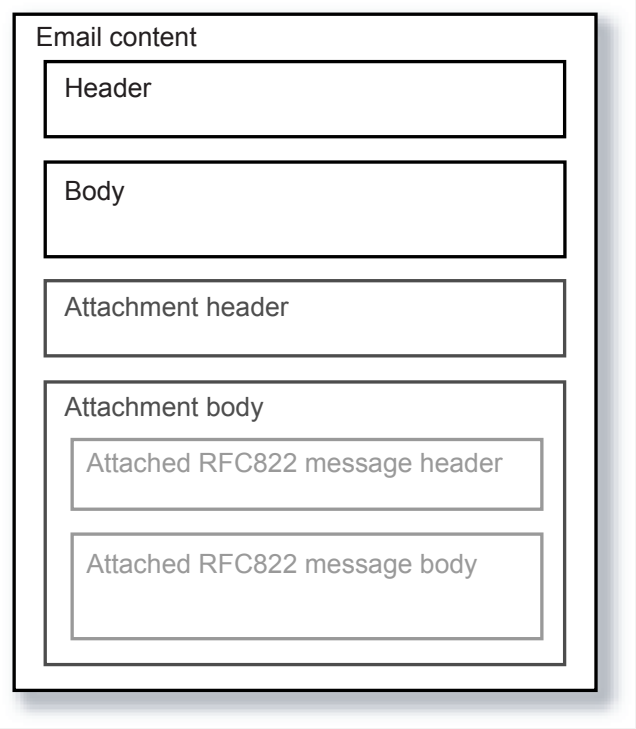


Figure: Internal Mail structure

Postfix Configuration and Administration
Controlling message flow
Checks

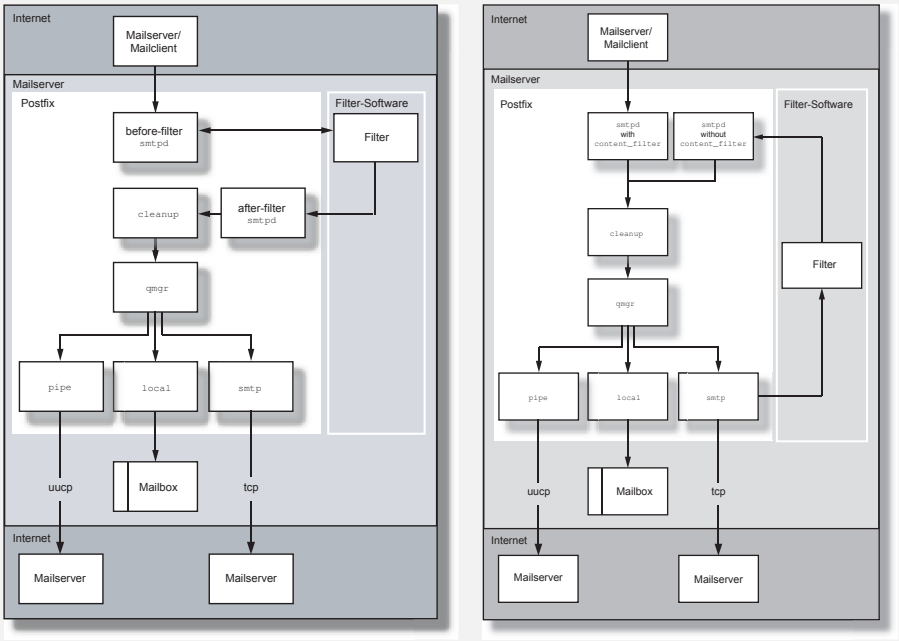
Checks work on content:

- Header
- Body
- MIME Header

Check functionality is limited on purpose. Postfix is not a content inspection engine.

The Postfix delegates the decision what to with the message to an external filter, either pre- or postqueue.

Pre- and postqueue filtering



(a) Prequeue filtering (b) Postqueue filtering

- `smtpd_milters`
- `non_smtpd_milters`

`smtpd_milters`

This is the most recent addition to Postfix. That way you can add the buggyness of Sendmail to Postfix.
No really, every milter I touched so far has been crap.

Postfix version 2.3 introduces support for the Sendmail version 8 Milter (mail filter) protocol.
This protocol is used by applications that run outside the MTA to inspect SMTP events (`CONNECT`, `DISCONNECT`), SMTP commands (`HELO`, `MAIL FROM`, etc.) as well as mail content.
All this happens **before** mail is queued.

The reason for adding Milter support to Postfix is that there exists a large collection of applications, not only to block unwanted mail, but also to verify authenticity (examples: Domain keys identified mail, SenderID+SPF and Domain keys) or to digitally sign mail (examples: Domain keys identified mail, Domain keys).
Having yet another Postfix-specific version of all that software is a poor use of human and system resources.
`http://sourceforge.net/projects/dkim-milter/`
`http://sourceforge.net/projects/sid-milter/`
`http://sourceforge.net/projects/dk-milter/`

Postfix version 2.4 implements all the requirements of Sendmail version 8 Milter protocols up to version 4, including message body replacement (body replacement is not available with Postfix version 2.3).