Port25™ Solutions, Inc.

PowerMTA™ v4.5 User's Guide

[Quick link to Directive Index](#)

TABLE OF CONTENTS

1. Introduction

1.1 General Description and Basic Functionality

PowerMTA is specialized, high performance Message Transfer Agent (MTA) software that intelligently and efficiently delivers large volumes of e-mail, allowing for maximum delivery and response. While all-purpose MTAs in use today perform a whole variety of tasks including delivering e-mail, these general solutions fall short both in regards to scalability and relevant feature sets, negatively affecting delivery rates and subsequent ROI. In contrast, PowerMTA was developed for this particular task, helping legitimate, permission based e-mail marketers, publishers, and service providers overcome the business and technology challenges of e-mail message delivery.

PowerMTA's basic functionality consists of receiving inbound or outbound messages via either standard e-mail protocols (SMTP) or programming interfaces, routing them based on either name services (DNS) or pre-configured information and delivering them using SMTP or locally to a program.

1.2 Minimum Requirements

PowerMTA currently runs on:
- Microsoft Windows 2008/2012 Intel and compatible processors
- Red Hat Enterprise Linux 6.0 or later on Intel and compatible processors (32 and 64 bit)
- Debian based Linux on Intel and compatible processors (32 and 64 bit)

Prerequisites:
- a name server (DNS), not necessarily on the same host,
- a static IP address and a domain name,
- a license activation key (LAK),
- minimum hardware requirements, depending on the size of the workload and mail queues, and the desired throughput rate.
- NTP (Network Time Protocol) – Clock must be synced to internet time server

Contact Port25 if you need hardware recommendations for your specific workload and throughput requirements.

1.3 Supported interfaces

PowerMTA supports the following submission and delivery interfaces:
- Submission via standard (extended) SMTP
- Submission via C/C++ API
- Submission via .NET API

- Submission via Perl API
- Submission via Java API
- Submission via "pickup" directory
- Delivery via standard (extended) SMTP
- Delivery to a local program

and will be supporting others in the future. Unless otherwise noted, these interfaces are available on all supported operating systems. You will find detailed information about the submission APIs in Chapter 7. Sample applications for the various APIs are also distributed along with PowerMTA.

1.4 Monitoring/Management Tools Summary

PowerMTA was designed to require only minimal configuration and very little management. However, for those who want to actively monitor and manage, a variety of flexible tools are available to do just that, both on a manual or automated basis. Full details and examples are provided in this Guide, but here is a quick list of the resources available:

- a built-in web-based monitor that allows you to view in real time what is currently taking place within the mailer. This monitor also includes a view and top level break down of the current mail queues, a view and breakdown of each VirtualMTA queue, a view of each job/campaign in the queue, as well as the last error received for each domain;
- a command line tool supporting a variety of monitoring and management commands, with three output formats (text, XML and DOM-style), which allow for easy integration into existing monitoring infrastructures or custom parsing and monitoring applications;
- detailed logging, both for incoming and outgoing connections, configurable both globally and for specific destination domains;
- detailed accounting data in XML format on each successful and unsuccessful transfer/delivery that can be post processed for delivery verification/accountability, for performance and throughput analysis, etc.
- a powerful accounting file search and reporting tool, allowing one to search for records in the accounting file and then to produce a custom output based on specific fields in the matching records.
- an accounting statistics application for extracting the most commonly needed information for monitoring and capacity planning.

2. PowerMTA Installation

2.1 Installing on Microsoft Windows

How to install:

1. Download the kit;

2. Log in as Administrator;

3. Open a command prompt as an administrator (right click and select run as administrator)

4. Run the PMC MSI from within this window

5. If you are installing PowerMTA for the first time or upgrading from a different version, copy the license key received from Port25 to a file named `license.dat` in the directory where PowerMTA was installed (e.g., `C:\pmta\license.dat`). Please ensure that the file is stored as in ASCII, *not* in Unicode;

6. Check that the configuration (e.g., `C:\pmta\config.dat`) suits your needs; the file can be edited with any ASCII-editor such as Notepad. See [Section 2.4](#) for details;

7. Start the mailer from the Services item in the Control Panel or by typing `net start pmta` in a Command Prompt window;

8. If you are upgrading from a v1.0 or v1.1 beta release and are using the C/C++ submission API, recompile and relink your applications;

9. If you intend to use the Perl API, install or update it from a command prompt window (possibly changing the path first) Currently only PERL 5.6 is supported:
   ```
   ppm install --location=C:\pmta\api Port25-Submitter
   ```

   and
   ```
   ppm install --location=C:\pmta\api Port25-Accounter
   ```

10. If you intend to use the Java API, include `pmta.jar` in your classpath. You may want to extract the JavaDoc API documentation from pmtajavadoc.zip. If you want to use PowerMTA with Sun's JavaMail API, install the provided `javamail.providers` file according to your system configuration. See the JavaMail documentation for details.

11. If you have an anti-virus and/or indexing service, they should not scan or handle the spool files, accounting files, or log files to prevent file locking issues with PowerMTA.

2.2 Installing on Linux

How to install:

1. Download the kit;

2. Log in as root;

3. If installing for the first time, optionally create a group called `pmta`. Membership to this group authorizes non-`root` users to execute PowerMTA commands and access the submission APIs. The group is created automatically by the installation procedures, but creating it manually allows you to also choose its numeric ID;

4. PowerMTA is built to work with a kernel as delivered with the Red Hat distributions. We do not encourage using a custom-built kernel. However, if using a custom-built kernel, please check Section 2.2.1 for how to generate a kernel with increased system limits;

5. If Goodmail was installed, remove it with:
```
# rpm -e PowerMTA-goodmail
```

6. Install the package (substituting the file name as appropriate) ;
```
# rpm -Uvh PowerMTA-4.*.rpm
```

   Or on Debian:
```
# dpkg -i PowerMTA-4.*.deb
```

7. If installing for the first time or upgrading from a different version of PowerMTA, copy the license key received from Port25;
```
# cp mykey.txt /etc/pmta/license
```

8. Check that the initial configuration (`/etc/pmta/config`) suits your needs, as described in Section 2.4;

9. Start the mailer (only necessary immediately after the install: it will be started automatically upon system startup);
```
# /etc/rc.d/init.d/pmta start
```

10. If you are upgrading from a v1.0 or v1.1 beta release and are using the C/C++ submission API, recompile and relink your applications;

11. If you intend to use the Perl Submission install or update it from `/opt/pmta/api`. The `Submitter` package is distributed in the standard `CPAN` format, so you will need to unpack, compile and install as follows.

```
# cd /opt/pmta/api
# tar -xzf Port25-Submitter-1.5.tar.gz
# cd Port25-Submitter-1.5
# perl Makefile.PL
# make test
# make install
```

12. If you intend to use the Java API, include `pmta.jar` in your classpath. You may want to extract the JavaDoc API documentation from pmtajavadoc.zip. If you want to use PowerMTA with Sun's JavaMail API, install the provided `javamail.providers` file according to your system configuration. See the JavaMail documentation for details.

2.2.1 System Limits on Linux

In many Linux kernels distributed from ftp://ftp.kernel.org the maximum number of processes per user (i.e., the `ulimit -u` limit) is 256, not allowing PowerMTA to start more than that many threads/connections. Additionally, the total number of processes in the system is limited to 512, so that if 256 are started by PowerMTA, you may find yourself unable to run any more commands.

To overcome these limitations, you can either use a kernel that has higher limits (e.g., Red Hat Linux 6.2 or later), or modify certain system constants and rebuild your kernel. If you prefer changing your current kernel,

1. edit the file `include/linux/tasks.h` and modify the constants
   o `NR_TASKS` (total maximum number of processes) to, for example, 2560
2. `MAX_TASKS_PER_USER` (maximum processes per user) to, for example, 2048

3. rebuild and install your kernel as usual

4. reboot

5. verify that `ulimit -u` displays the new limit

2.3 Initial Configuration

There are only two things that *must* be manually defined in the configuration file, in addition to the supplied default entries, in order to have PowerMTA up and running: the postmaster address(es) and the relaying control configuration.

Besides being the traditional address for which external parties can reach the maintainer of the mail server, the postmaster address is used by PowerMTA and its watchdog monitoring facility, `pmtawatch`, to send crash reports and other information to upon successful automatic restart of the mailer (if it crashes).

12

For security reasons, PowerMTA does not allow external relaying by default; the default configuration only allows messages to be sumitted from the local 127.0.0.1 IP address. In order to submit messages for delivery from a different host, you will need to enter its IP address(es) into the configuration file by creating one or more `source` entries and specifying the `always-allow-relaying` directive.

Note that `relay-domain` and `relay-address` directives in the configuration file are really only useful when PowerMTA is used to handle inbound traffic. That allows you to specify the domains in messages which PowerMTA is to accept from any sources. When configuring PowerMTA for outbound, you want it to accept e-mail from your feeders for all domains, by using `always-allow-relaying`.

See Chapter 3 for detailed information on the configuration file.

2.3.1 IPv6

PowerMTA now supports IPv6. This requires an OS that supports IPv6 and an IPv6 address. Other than that, the IPv6 address can be used in place of an IPv4 address in the configuration file.

Examples:

```
smtp-listener [FE80::0202:B3FF:FE1E:8329:25]:25

<domain test.port25.com>
    route [2001:db8::1]:26
</domain>

<source fe80::202:b3ff:fe1e:1839>
  smtp-service yes
</source>

http-access ::1 admin
```

**The following should only be used if the kernel reports different outputs for 'ifconfig' and 'ip addr show'.

On linux, if the system supports it, PMTA uses the same method as 'ip addr show' to discover IP addresses. There is support within PMTA for using the 'ifconfig' way of discovering IP addresses. To use this method, they should edit the '/etc/init.d/pmta' file and uncomment the line:

```
export PMTA_USE_GETIFADDRS=0
```

The line is commented by default. When the PMTA service is started after uncommenting the line above, PMTA will be forced to use the same method as 'ifconfig' does when discovering IP addresses on the system.

2.4 Uninstalling

On each platform PowerMTA is installed with the platform's standard packaging software. You can uninstall PowerMTA using the platform's existing uninstall option:

- On Windows, open the Control Panel, open Add/Remove Programs, select the PowerMTA item, and click on Add/Remove. After confirming that you would like to proceed, that will initiate the uninstall process.
- On Linux, log in as root and execute rpm -e PowerMTA

Any configuration, logging, accounting and message files found in PowerMTA's directories will not be automatically removed — so neither will the directories themselves. On Linux the uninstall software will display a list of the remaining files and the commands you need to execute to remove them. On Windows, you can simply remove the complete directory tree onto which you had installed PowerMTA.

2.5 Running PowerMTA on a Virtual Server

While this is technically possible, it is not a recommended setup unless the virtual server has adequate hardware and the virtual server software is configured correctly. The sharing of resources on a mis-configured virtual machine may create an environment under load that could cause PowerMTA to become unresponsive.

If PowerMTA is going to be run on VMware®, please take the following steps to ensure that PowerMTA gets the needed resources:

- Open your vSphere client
- Right click on the PowerMTA server and select "Edit Settings"
- Select the "Resources" tab
- Change the shares value for CPU/Memory/Disk to the following values:
  CPU = 1000000
  Memory = 1000000
  Disk = 4000
- For CPU, Memory, and Disk also make sure the "Unlimited" checkbox is checked for each in the same section
- Restart the PowerMTA server

These can be changed again later if needed or the settings can be turned down to find the optimal setting, but we have found that this helps ensure PowerMTA runs properly in a virtual environment.

3. PowerMTA Configuration

3.1 Working with the Configuration File

PowerMTA's configuration file is a straightforward ASCII text file that can be read and modified with most text editors (e.g., Notepad, vi, emacs). Settings are entered one per line and are formed by a keyword on the left identifying a directive and a value on the right, as in the example below:

```
smtp-port 2525   # use alternate SMTP port
```

Certain directives, like those applying to a specific destination domain, are grouped based on what they apply to:

```
<domain test.port25.com>        # 'test' host
    route [127.0.0.1]:11111
    log-commands yes
</domain>
```

As indicated in the examples above, comments can be introduced by prefixing the comment with a hash mark "#". Like other file formats using the hash mark for comments, PowerMTA ignores the "#" and any characters following it until the end of the line.

PowerMTA reads the configuration file upon startup. After making changes to the configuration file, you must thus either re-start it or execute the `pmta reload` command. Because it is more efficient and does not cause an interruption in service, in general the `reload` command is preferrable to re-starting PowerMTA. However, some directives such as the location of the spool directories cannot be reloaded dynamically and require a restart. Unless otherwise noted in the descriptions below, you can assume the directives to be dynamically reloadable.

PowerMTA supports the ability to remotely modify/upload a configuration file via the web UI (and thus via HTTP). To push the configuration programmatically, one needs to issue a HTTP POST, passing the configuration file in the field named "file" to the URL http://127.0.0.1:8080/editConfig. One also needs to be authorized for "admin" access via the http-access directive.

3.2 Configuration Directives Categories and Tags

The following sections describe the various configuration directives available, grouped by function. Each description includes a little table specifying the directive's *scope*, *type*, *attributes* and *default*.

3.2.1 Directive "Scope" Defined

A directive's scope indicates where in the configuration file it can be specified.

3.2.2 Directive "Type" defined

A directive's *type* indicates the kind of values accepted. They are as follows:

boolean

> A boolean directive, accepting the values `yes`, `true`, `no` and `false`.

number

> A number. The valid ranges depend on how the number is used. For example, port numbers must be between 0 and 65535 and may be subject to further limitations in range by the operating system.

file name

> A file's name. Unless otherwise noted, should always be an absolute file name, either beginning at the root (on Unix) or at the drive name (Windows).

directory name

> A directory's name. Unless otherwise noted, should always be an absolute specification, either beginning at the root (on Unix) or at the drive name (Windows).

time interval

> A sequence like `1d2h3m4s`, meaning one day, two hours, three minutes and four seconds. Parts of the specification can be omitted, like in `4d12h`, which stands for four days and twelve hours.

IP address

> An IP address in *dotted decimal notation*, like `10.0.0.1`.

IP address list

> The directive builds a list of IP addresses. Each directive specifies an additional IP address that is appended to the list.

e-mail address

> An e-mail address, like `support@port25.com`, *not* including a free form name like "John E. Doe" or <> characters, etc.

string

> A string of characters. If the string contains any spaces or tabs, it should be enclosed in double quotes ("").

directory name list

> The directive builds a list of directory names. Each directive adds a directory name to the end of the directory list. Unless otherwise noted, should always be an absolute specification, either beginning at the root (on Unix) or at the drive name (Windows).

IP address list

> The directive builds a list of IP address ranges. This list is then used by PowerMTA to match IP addresses. For example, when deciding whether to grant access to the web-based monitor, PowerMTA scans the list built with the `http-access` directive.

16

Each directive specifies either an IP address in dotted decimal notation (like `10.0.0.1`), or a range of IP addresses in CIDR notation (like `10.0.0.0/8`). CIDR specifications are formed by an IP address prefix and by the number of significant bits within the 32-bit number formed by the prefix. In the example above, only the first eight bits (the `10` portion) are significant, so any IP addresses starting with `10.` would match.

domain

The directive builds a list domain wildcards. This list is then used by PowerMTA to match domain names.

Domains can be specified in one of the following formats:

`*`

Matches all domains.

`domain`

A single, fully qualified domain name. Matches that specific domain only.

`*.domain`

Matches any subdomains of *domain*, but *not* the domain itself.

`[*.]domain`

Matches *domain* as well as its subdomains.

`[ip/mask]`

Matches domains in the dotted decimal format, like `[127.0.0.1]`. The actual IP address in the domain must match the given CIDR specification (*ip* prefix and *mask* number of significant bits). `[0/0]` matches all domains in dotted decimal format.

3.2.3 Directive "Attribute" defined

A directive's *attributes* indicate

required/optional

whether a directive must be specified in the configuration file

obsolete

whether a directive is obsolete and should no longer be used

non reloadable

whether a directive can be reloaded dynamically by using the `pmta reload` command. If a directive is non reloadable, PowerMTA must be restarted before any changes to it become active.

3.2.4 Directive "Default" defined

A directive's *default* is simply the value or behavior when the directive is *not* specified in the configuration file. Please note that the initial configuration may differ from the directive's default.

3.2.5 <global> Directives Defined

The directive is "global" and affects PowerMTA as a whole.

3.2.6 <domain> Directives Defined

The directive affects items (like e-mail queued, outgoing connections, etc.) related to the specified domain name. The directive should be specified within a <domain> entry, like in

```
<domain *.port25.com>
    bounce-after 6d
</domain>
```

Domains can be specified in one of the following formats:
*

Matches all domains.
domain

A single, fully qualified domain name. Matches that specific domain only.
*.domain

Matches any subdomains of domain, but not the domain itself.
[*.]domain

Matches domain as well as its subdomains.
[ip/mask]

Matches domains in the dotted decimal format, like [127.0.0.1]. The actual IP address in the domain must match the given CIDR specification (ip prefix and mask number of significant bits). [0/0] matches all domains in dotted decimal format.

Domain directives may be added in any order to the configuration file. When PowerMTA needs to determine what parameters to use for a specific domain, it uses the *best* matching (most specific) entry, regardless of the order domain directives are placed into the configuration file.

```
<domain discard.port25.com>  # matches 'discard' only
    ...
</domain>
<domain *.port25.com>        # matches subdomains of port25.com
    ...
</domain>
<domain [*.]port25.com>   # matches port25.com and all subdomains
    ...
</domain>
<domain *>                   # matches all domains
    ...
</domain>
```

When PowerMTA reads the configuration file, it merges settings for related domains from more general to more specific domains, so you can provide defaults for all domains by specifying them for the "*" pseudo-domain or for a specific site by specifying them for the toplevel domain for the site.

In the following example, commands will be logged for all proper subdomains of port25.com (but not for port25.com itself) and undelivered messages to any address at port25.com will bounce after four days, except for messages to discard.port25.com, which will bounce after one hour.

```
<domain discard.port25.com>
    bounce-after 1h
</domain>
<domain *.port25.com>
    log-commands yes
</domain>
<domain [*.]port25.com>
    bounce-after 4d
</domain>
<domain *>
    bounce-after 6d
</domain>
```

To include domains that have different root domains (e.g. yahoo.com & yahoo.co.uk), PowerMTA has the ability to define top level domain macros. To use this feature, you would first define a domain macro with the desired list of bindings.

```
domain-macro topLevel com, de, co.uk
```

and then use it in the <domain> tag like.

```
<domain aol.$topLevel>
   ...settings...
</domain>
```

PowerMTA automatically expands that to aol.com, aol.de, and aol.co.uk. You can also use several macros within a <domain>, as in the following:

```
domain-macro topLevel com, de, co.uk
domain-macro aol aol, aim

<domain $aol.$topLevel>
   ...settings...
</domain>
```

This expands to aol.com, aim.org, aol.de, aim.de, aol.co.uk and aim.co.uk.

You can also use macros within macros:

```
domain-macro com com, co.uk, com.br
domain-macro tld $com, org, net
```

In which case tld expands to com, co.uk, com.br, org and net. Any loops are detected and result in an error (which prevents the configuration from being reloaded).

Also, specifically listed domains always take precedence over expanded ones. For example, in.

```
<domain aol.com>
    max-smtp-out 100
</domain>

domain-macro com com, co.uk

<domain aol.$com>
    max-smtp-out 50
</domain>
```

aol.com would have a limit of 100.

3.2.7 <source> Directives Defined

The directive affects incoming SMTP connections from the given source. The directive should be specified within a `<source>` entry, like in

```
<source 10.0.0.0/8>
    log-connections yes
</source>
```

Per-source directives can be specified in two ways: by IP address and by name.

Each `<source>` entry to be applied by IP address specifies either an IP address in dotted decimal notation (like `10.0.0.1`), or a range of IP addresses in CIDR notation (like `10.0.0.0/8`). CIDR specifications are formed by an IP address prefix and by the number of significant bits within the 32-bit number formed by the prefix. In the example above, only the first eight bits (the `10` portion) are significant, so any IP addresses starting with `10.` would match.

A `<source>` entry to be applied by name specifies its name in the entry, like in `<source S1>`.

Source entries are scanned sequentially by PowerMTA when a new incoming connection request is received. The *first* matching entry for any directive is used, so it is important to enter them in the proper order — from more specific to more general, as in the example below:

```
<source 10.0.0.5>       # matches 10.0.0.5 only
    ...
</source>
<source 10.0.0.0/8>     # matches 10.*
    ...
</source>
<source 0/0>            # matches all
    ...
</source>
```

Note that PowerMTA merges ("inherits") settings in `source` entries like it does for the `domain` entries, allowing you to provide defaults in a more generic entry (such as `0/0`). Still, contrary to the `domain` entries, `source` entries cannot be automatically sorted, which is why you need to specify them in the order you wish them used.

Named entries are used to override the settings obtained based on the source IP address. They apply only when specifically requested, like in the `source=...` parameter of the `smtp-listener` directive.

If using a pickup directory, the format is <source {pickup}>.

<source {pickup}> and <source {auth}> are reserved and the only two <source> tags that should use { & }.

3.2.8 <source-group> Directives Defined

The directive affects incoming SMTP connections from the given group of `<source>` tags. The directive allows for specifying certain source configuration items whose functionality require them to be grouped by name. The shared resources will be defined in the `<source-group>` tag. To add a `<source>` to the shared resource, specify the name of the `<source-group>` in the `<source>` tag. An example would look similar to the following:

```
<source-group localSources>
    max-smtp-in 10
    reserved-smtp-in 5
</source-group>

<source 192.168.0.30>
    smtp-service yes
    always-allow-relaying yes
    source-group localSources
</source>

<source 192.168.0.40>
    smtp-service yes
    always-allow-relaying yes
    source-group localSources
</source>
```

21

Each `<source>` that defines the source-group directive of the same name will be controlled by the settings in the named `<source-group>` tag.

These directives are reloadable.

3.2.9 <virtual-mta> Directives Defined

The directive specifies behavior specific to a "VirtualMTA". VirtualMTA directives only apply when that VirtualMTA has been selected through one of the supported selection methods. They should be specified in a `<virtual-mta>` entry, like in

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 mta1.company.com
    <domain *>
      dkim-sign yes
    </domain>
</virtual-mta>
```

See Chapter 8 for more information on VirtualMTAs.

3.2.10 <virtual-mta-pool> Directives Defined

The directive is valid in a `virtual-mta-pool` entry. VirtualMTA pools are groups of VirtualMTAs. They share the same name space as VirtualMTAs, so that you can select a pool instead of a specific VirtualMTA. When you select a VirtualMTA pool, the VirtualMTAs in that pool are used in round-robin fashion to deliver the message. See Chapter 8 for more information on VirtualMTAs and VirtualMTA pools.

A <domain> may be defined inside of a <virtual-mta-pool>. If this is done, the defined directives will be inherited by any VirtualMTA that is part of the pool. If a conflicting directive is defined directly in the given VirtualMTA, the directive in the VirtualMTA overrides the one defined in the <virtual-mta-pool>.

The following directives are also supported:

- cold-virtual-mta

- domain-key

- max-smtp-out

- max-smtp-msg-rate

- include-headers-from

22

3.2.11 <pattern-list> Directives Defined

The directive adds a regular expression pattern to a list of patterns. Such patterns can be used to select a VirtualMTA based on the message's originator. It should be entered in a `<pattern-list>` entry, like in

```
<pattern-list myList>
  mail-from /^newsletter/           virtual-mta=newsletters
  mail-from /^specials/             virtual-mta=specials
  rcpt-to   /^stud.*@somesite.com$/ virtual-mta=students
  rcpt-to /password-reset@customer.com/ recipient-priority=60
  rcpt-to /special@customer.com/ virtual-mta=high, recipient-priority=60
</pattern-list>
```

Conditional AND/OR style patterns can be defined. See Chapter 8 for more information on selecting VirtualMTAs. Section 3.2.16 describes pattern lists in greater detail.

3.2.12 <smtp-pattern-list> Directives Defined

The directive adds a regular expression pattern to a list of patterns. Such patterns can be used to have PowerMTA take certain actions based on the SMTP replies received from the remote mailer. It should be entered in a `<smtp-pattern-list>` entry, like in

```
<smtp-pattern-list myList>
    reply /blocked/  mode=backoff
    reply /permanently deferred/ bounce-queue
    reply /too many connections/ skip-mx
    reply /account over quota/ bounce-rcpt
</smtp-pattern-list>
```

Section 3.2.36 describes SMTP pattern lists in greater detail.

3.2.13 \<bounce-category-patterns> Directives Defined

This directive makes the bounce categories used in CSV accounting files and the X-PowerMTA-BounceCategory DSN field customizable. A customer can specify the tag in the configuration file, in which case the patterns specified there take precedence over those defined in config-defaults file. The data in the `<bounce-category-patterns>` tag consist of two fields. The first field is the SMTP response on which to match, and the second field is the name of the category to be used. Custom category names are allowed. It should be entered in a `<bounce-category-patterns>` entry, like in

```
<bounce-category-patterns>
    /spam/ spam-related
    /no longer (valid|available)/ bad-mailbox
    /mailbox +(is +)?full/ quota-issues
    /banned/ policy-related
</bounce-category-patterns>
```

`<bounce-category-patterns>` uses pattern matching in the same method as does `<pattern-list>`. Section 3.2.16 describes SMTP pattern lists and pattern matching in greater detail.

3.2.14 \<acct-file> Directives Defined

These directives only apply to the CSV accounting file. The directive specifies behavior specific to the CSV accounting file. It should be used in a manner similar to the following:

```
<acct-file log\acct.csv>
   move-interval 5m
   max-size 50M # MB
   sync no
</acct-file>
```

See Chapter 11 for more information on the accounting file.

24

3.2.15 <smtp-user> Directives Defined

The directive adds the abilty to specify a given user who should be able allowed to send email, regardless of the IP from which the mail is sent, to PowerMTA for relaying. The directive should be used once per user and reference a general source tag that controls the various directives to control the messages. It should be entered in a `<smtp-user>` entry, like in

```
<smtp-user jsmith>
  password MyPassword
  source {auth}
</smtp-user>

<smtp-user jsmith@example.com>
  password MyPassword
  source {auth}
</smtp-user>

<source {auth}>
    always-allow-relaying yes  # allow feeding for defined users
    process-x-virtual-mta yes  # allow selection of a VirtualMTA
    max-message-size 0         # 0 implies no cap, in bytes
    smtp-service yes           # allow SMTP service
    require-auth true
</source>

<source 0/0>
    allow-unencrypted-plain-auth yes
</source>
```

Section 10.2.2 describes smtp-user in greater detail.

As a note, <source {auth}> is reserved, and if no source for <smtp-user> is specified, <source {auth}> will be used. Any other non-reserved named source may be used for <smtp-user>.

25

On a Linux the system accounts may be used. To do so configure the "authentication-method system" directive similar to the following:

```
<smtp-user jsmith>
  authentication-method system
  source {auth}
</smtp-user>

<source {auth}>
    always-allow-relaying yes  # allow feeding for defined users
    process-x-virtual-mta yes  # allow selection of a VirtualMTA
    max-message-size 0         # 0 implies no cap, in bytes
    smtp-service yes           # allow SMTP service
    require-auth true
</source>

<source 0/0>
    allow-unencrypted-plain-auth yes
</source>
```

3.2.16 <spool> Directives Defined

The directive allows one to specifiy the location of the spool file. New spools may be activated with a reload of PowerMTA. Deactivating a <spool> requires a restart of PowerMTA.

```
<spool /var/spool/pmta>
  delete-file-holders yes
  deliver-only yes
</spool>

<spool C:\pmta\spool>
</spool>
```

3.2.17 <bounce-processor> Directives Defined

The directive allows for using the bounce processor in PowerMTA. If configured, PowerMTA will look for bounce emails when processing incoming messages.

```
<bounce-processor>
      deliver-unmatched-email yes        # default: no
      deliver-matched-email yes          # default: no
      forward-unmatched-to auto-feedback@port25.com
      forward-errors-to auto-feedback@port25.com
   <address-list>
      domain domain.to.filter            # whole domain
      address /regex@domain.to.filter/   # regex
   </address-list>
</bounce-processor>
```
See Section 12.1.6 for more detail on the <bounce-processor> directives.

3.2.18 <feedback-loop-processor> Directives Defined

The directive allows for using the feedback loop processor in PowerMTA. If configured, PowerMTA will look for feedback loop emails when processing incoming messages.

```
<feedback-loop-processor>
      deliver-unmatched-email no
      deliver-matched-email yes          # default: no
      forward-unmatched-to auto-feedback@port25.com
      forward-errors-to auto-feedback@port25.com
       <address-list>
          address /fbl@mydomain.com/
      </address-list>
</feedback-loop-processor>
```
See Section 12.1.7 for more detail on the <feedback-loop-processor> directives.

3.2.19 <address-list> Directives Defined

The directive defines the address or patterns to match on when using <bounce-processor> or <feedback-loop-processor>. See Section 12.1.6 for more detail on the <bounce-processor> directives. See Section 12.1.7 for more detail on the <feedback-loop-processor> directives.

3.2.20 <aliases> Directives Defined

There may be a need to use aliases or to forward emails. This directive allows a list of email addresses to be defined for which PowerMTA should forward messages to different addresses. For example, if a message is received by PowerMTA for example@domain1abc.com, then forward the message on to support@port25.com.

Alias expansion does not change VMTA assignment. VMTA assignment using pattern-list is done first, then when the recipient is being routed to its queue, alias expansion is applied.

See Section 12.9 for more detail on the <aliases> directives.

3.2.21 <mx-rollup-list> Directives Defined

Allows one to define "rollup" queues based on the MX records of the recipient domain, in order to consolidate separate but related recipient domains into one queue.

3.3 Configuration Directives

3.3.1 General Directives

postmaster

> **Scope:** global
>
> **Type:** e-mail address
>
> **Attributes:** optional
>
> **Default:** none

Specifies an e-mail address for the person responsible for PowerMTA's operation. This should *always* be specified because it is used by the internal watchdog facility for crash and LAK expiry notifications. Messages sent to the local `postmaster` and `abuse` addresses are also by default forwarded to the given addresses. You can specify more than one address by entering `postmaster` multiple times.

Example:

```
postmaster you@your.domain
postmaster you@your.domain.at.home
```

3.3.2 Message Spool Directives

spool

        Scope: global

        Type: directory name list

        Attributes: required, non reloadable

        Default: none

Deprecated, use <spool> tag and directives.

Specifies an additional directory for the message spool. The message spool is where PowerMTA stores the message files while queued for delivery. At least one `spool` entry is required. Additional entries can be added for greater performance, for example by distributing the I/O load over several physical disks.

Example: (Unix)

```
# distribute PowerMTA's spool I/O load away from the system
disk
# and over to disks 2 - 4
spool /dsk2/pmta-spool
spool /dsk3/pmta-spool
spool /dsk4/pmta-spool
```

Example using <spool> tag: (Unix)

```
<spool /dsk2/pmta-spool>
</spool>
<spool /dsk3/pmta-spool>
</spool>
<spool /dsk4/pmta-spool>
</spool>
```

delete-file-holders

        Scope: spool

        Type: boolean

        Attributes: optional

        Default: false

Determines whether PowerMTA should delete any message holders in the spool directory after a message has been delivered. Will remove 0kb files at the cost of some HDD I/O. Empty folder will also be removed.

`deliver-only`

> **Scope:** spool
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

Determines whether PowerMTA should stop adding new message files to a spool location. Messages from the spool location will however continue to be delivered. This can be usefull if a spool was added from another server for recovery, but only for purposes of delivery.

Note: For a spool directory defined with the deprecated spool directive, 'deliver-only' and 'delete-file-holders' will default to false, and cannot be changed.

`spool-delete-corrupted`

> **Scope:** global
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

Determines whether PowerMTA should delete any message files in the spool found to contain encoding errors, such as those caused by disk data corruption.

`spool-max-files`

> **Scope:** global
> **Type:** `auto` or number
> **Attributes: deprecated**, optional, non reloadable
> **Default:** auto

This directive is currently an alias for `spool-max-recipients`. Please use `spool-max-recipients` instead.

`min-free-space`

> **Scope:** spool
> **Type:** {n{B|K|M|G|T}}
> **Attributes:** optional
> **Default:** 0

If the number of bytes available (to non-root users) falls below what this directive specifies (in MB), PowerMTA stops accepting email. When available disk space increases above that level, messages are accepted again.

`spool-max-recipients`

       Scope: global

       Type: `auto` or number

       Attributes: optional, reloadable

       Default: auto

Specifies the maximum number of recipients to allow in the spool, to help prevent overwhelming PowerMTA with more messages than it has the resources to handle. When this limit is reached, feeding new messages into PowerMTA fails with a transient error message. If set to `auto`, the maximum number is determined automatically based on system resources (currently only on RAM, but this may change in the future).

3.3.3 Relaying Control Directives

`always-allow-relaying`

       Scope: source

       Type: boolean

       Attributes: optional

       Default: false

Specifies whether SMTP senders from the given source should be allowed to relay e-mail through PowerMTA, even if the recipients are not for one of the `relay-domain`s. Typically you should enable this in the `source` entries of "internal" hosts feeding PowerMTA through SMTP.

relay-domain

    Scope: global

    Type: domain

    Attributes: optional

    Default: local domains

Domain name (and domain wildcards) for which PowerMTA will accept e-mail. Any host is allowed to relay e-mail for these domains, independent of its source IP address. Directive may be specified more than once.

This directive is mostly useful for handling inbound e-mail. When setting up PowerMTA for relaying inbound e-mail to some other host, you should enter the corresponding domain name(s) into the relay domains list. Conversely, if in your installation PowerMTA is handling outbound e-mail only, there is no need to configure any relay domains.

Example:

```
# route e-mail for other.host to 10.0.0.5:1025
<domain other.host>
    route [10.0.0.5]:1025
</domain>

# allow relaying for other.host
relay-domain other.host
```

relay-address

    Scope: global

    Type: string

    Attributes: optional

    Default: none

Email address for which PowerMTA will accept e-mail. Any valid email is allowed to relay e-mail for independent of its source IP address. Directive may be specified more than once.

This directive is mostly useful for handling inbound e-mail. When setting up PowerMTA for relaying inbound e-mail to some other host.. Conversely, if in your installation PowerMTA is handling outbound e-mail only, there is no need to configure any relay addresses.

Example:

```
relay-address bounce@bounce.yourdomain.com
relay-address /@foo(bar)?\.com$/
```

33

alias

> **Scope:** alias
> **Type:** string
> **Attributes:** optional
> **Default:** none

> This directive allows a list of email addresses to be defined for which PowerMTA should forward messages to different addresses. For example, if a message is received by PowerMTA for example@domain1abc.com, then forward the message on to support@port25.com. See Section 12.7 for more information.

relay-debug

> **Scope:** global
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

> Specifies that relaying control is to be performed in "debugging" mode. Normally, recipients to which relaying is not allowed are rejected with persistent (5XX) SMTP error codes, causing immediate bounces. In debugging mode, a transient (4XX) error is returned instead to the sending mailer, allowing it to continue trying to deliver to that recipient. This directive comes handy when making changes to the relaying control configuration since it gives you a chance to review the log for any unintended rejections and correct the configuration before messages are bounced.

deliver-unmatched-email

> **Scope:** bounce-processor, feedback-loop-processor
> **Type:** boolean
> **Attributes:** optional
> **Default:** no

> Determines whether any emails that were not matched by bounce/feedback-loop processors should be delivered. If set to false, unmatched emails are discarded.

```
deliver-matched-email
```
        Scope: bounce-processor, feedback-loop-processor

        Type: boolean

        Attributes: optional

        Default: no

Determines whether any emails that were matched by bounce/feedback-loop processors should be delivered. If set to false, matched emails are discarded.

```
deliver-email
```
        Scope: bounce-processor, feedback-loop-processor

        Type: boolean

        Attributes: optional

        Default: no

Deprecated. Use deliver-matched-email/deliver-unmatched-email instead.

```
forward-unmatched-to
```
        Scope: bounce-processor, feedback-loop-processor

        Type: email address

        Attributes: optional

        Default: false

This directive allows for forwarding messages unmatched by the bounce processor or feedback loop processor, to facilitate collecting emails that matched the defined pattern or email, but were not able to be determined as a bounce or feedback loop email.

```
forward-errors-to
```
        Scope: bounce-processor, feedback-loop-processor

        Type: email address

        Attributes: optional

        Default: false

This directive allows for forwarding messages matched by the bounce processor or feedback loop processor, to facilitate collecting emails that matched the defined pattern or email, but there was an error in processing the message.

address

     Scope: address-list

     Type: string, email address

     Attributes: optional

     Default: false

This directive takes a regular expression (delimited by //) to be matched against the entire RCPT TO email address. If the pattern is matched, the message is passed to the bounce or feedback loop processor. See Section 12.1.6 for more detail on the <bounce-processor> directives. See Section 12.1.7 for more detail on the <feedback-loop-processor> directives.

address-file

     Scope: address-list

     Type: string

     Attributes: optional

     Default: false

Iincludes into the <address-list> all addresses in the file, one per line.

domain

     Scope: address-list

     Type: string, domain

     Attributes: optional

     Default: false

This directive takes a regular expression (delimited by //) to be matched against the domain portion of the RCPT TO email address. If the pattern is matched, the message is passed to the bounce or feedback loop processor. See Section 12.1.6 for more detail on the <bounce-processor> directives. See Section 12.1.7 for more detail on the <feedback-loop-processor> directives.

3.3.4 SMTP Service Directives

add-date-header

      Scope: source

      Type: {no|yes|override}

      Attributes: optional

      Default: no

Specifies whether PowerMTA should add a `Date` header if missing. If a `Date` header is present, it is not overridden unless using the override parameter.

Note: automatic adding of the `Date` header is not supported for mailmerge messages. When submitting mailmerge messages, you can use the `*date` variable to easily accomplish the same.

date-header-time

      Scope: source

      Type: {reception|delivery}

      Attributes: optional

      Default: reception

Specifies whether PowerMTA should set the `Date` header to the time the message was received and queued for delivery, or the time the message was actually delivered. The directive add-date-header needs to be set to yes or override when using this directive.

bcc

      Scope: source

      Type: sting

      Attributes: optional

      Default:

This directive takes an email address as a value, and if set, causes each message received from that source to be BCC'ed to the given email address. This can be used for creating a copy of every message for archival purposes.

add-message-id-header

> **Scope:** source
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

Specifies whether PowerMTA should add a `Message-Id` header if missing. If a `Message-Id` header is present, it is not overridden.

add-received-header

> **Scope:** source
> **Type:** boolean
> **Attributes:** optional
> **Default:** true

Specifies whether PowerMTA should add a `Received` header upon reception of an email.

run-as-root

> **Scope:** global
> **Type:** boolean
> **Attributes:** optional, non reloadable
> **Default:** true

Specifies whether PowerMTA should run as root or as user pmta. Only works on non-Windows systems.

allow-auth

> **Scope:** source
> **Type:** boolean
> **Attributes:** optional
> **Default:** true

Specifies whether PowerMTA should allow usage of the AUTH SMTP command.

38

`remove-header`

> **Scope:** source, domain
>
> **Type:** string
>
> **Attributes:** optional
>
> **Default:**

Specifies a header that PowerMTA should remove from the incoming or outgoing message. The directive takes a comma separated list of headers if there is more than one header to be removed. The header is removed at delivery, not when the message is queued to disk. The directive may be used in a <source> or <domain>, but is not required to be defined in both. To indicate no header is to be removed use "" (only needed when overriding an inherited setting).

> **Example:**
>
> ```
> <source 127.0.0.1>
>     remove-header nameOfYourCustomHeader,nameOfOtherHeader
> </source>
>
> <domain example.com>
>     remove-header nameOfYourCustomHeader,nameOfOtherHeader
> </domain>
>
> <domain *>
>     remove-header ""
> </domain>
> ```

`require-auth`

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Allows for requiring authentication before any emails are received via SMTP (local or not).

`allow-mailmerge`

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should allow usage of the mailmerge SMTP extensions from the given source.

39

allow-unencrypted-plain-auth

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   false

> Specifies whether PowerMTA should allow usage of the PLAIN SASL authentication mechanism in unencrypted connections (PowerMTA does not currently support encryption of connections).

allow-starttls

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   no
>
> This directive allows for using server-side STARTTLS.

require-starttls-before-auth

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   no
>
> Allows for requiring STARTTLS before any AUTH commands are issued.

smtp-server-tls-certificate

> **Scope:**   global
>
> **Type:**   string
>
> **Attributes:** optional
>
> **Default:**
>
> Allows for setting the file and password to be used for server-side STARTTLS. Takes a file name and the certificate's password as arguments. See http://www.openssl.org/ for information on creating certificates.

> **Example:**

```
smtp-server-tls-certificate /etc/pmta/yourcert.pem Password
```

`smtp-server-tls-ca-file`

> **Scope:** global
> **Type:** string
> **Attributes:** optional
> **Default:**

Specifies the location of a local text file containing one or more certificates in a chain that will be presented to clients by PowerMTA when they connect to PowerMTA. Primarily used by submitters to authenticate PowerMTA when using TLS based submission.

> **Example:**

```
smtp-server-tls-ca-file /etc/pmta/certchain.pem
```

`password`

> **Scope:** smtp-user
> **Type:** string
> **Attributes:** optional
> **Default:**

Specifies the password for the defined user in <smtp-user>..

`authentication-method`

> **Scope:** smtp-user
> **Type:** system
> **Attributes:** optional
> **Default:**

Specifies the corresponding Linux user account should be used for authentication.

`source`

> **Scope:** smtp-user
> **Type:** string
> **Attributes:** optional
> **Default:**

Specifies the <source> to be used for the defined user in <smtp-user>..

broken-auth-clients

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** true

Changes the manner in which PowerMTA implements AUTH to allow for some email clients, such as Outlook or Outlook Express.

verp-default

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** no

If you enable VERP by setting verp-default to yes or true, PowerMTA will create a customized SMTP MAIL FROM address for the message based on the original SMTP MAIL FROM address and the RCPT TO addresses in the message. Since one can then identify the original RCPT TO address in this new encoded SMTP MAIL FROM address, this may make tracking bounces easier for you.

The format is fixed, and can be see in the following example;

Original SMTP MAIL FROM: jsmith@example.com
SMTP RCPT TO: jdoe@yahoo.com
VERPed SMTP MAIL FROM: jsmith-jdoe=yahoo.com@example.com

check-domainkeys-inbound

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false
>
> Deprecated

Specifies whether to perform a DomainKeys check. If the message has DomainKeys and it passes, PowerMTA will add an Authentication-Results header stating the passing results.

check-dkim-inbound

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   false

Specifies whether or not PowerMTA should enable checking DKIM signatures on inbound messages. If the message has DKIM and it passes, PowerMTA will add an Authentication-Results header stating the passing results.

trace-domainkeys-check

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   false
>
> Deprecated

Allows selecting whether the trace data (currently including all the DNS data looked up) is included in the Authentication-Results headers for the domainkey check.

trace-dkim-check

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   false

Allows selecting whether the trace data (currently including all the DNS data looked up) is included in the Authentication-Results headers for the dkim check.

trace-mfrom-check

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   false

Allows selecting whether the trace data (currently including all the DNS data looked up) is included in the Authentication-Results headers for the mfrom check.

```
max-errors-per-connection
```
Scope: domain

Type: number

Attributes: optional

Default: unlimited

This directive tells PowerMTA to break a connection after a certain amount of recipient level errors. The default is unlimited, meaning that PowerMTA will not break a connection due to a number of recipient level errors.

```
check-mfrom-outbound
```
Scope: domain

Type: boolean

Attributes: optional

Default: false

Specifies whether PowerMTA should check outbound messages' "mfrom" (domain in the `MAIL FROM` or `HELO` SMTP commands) for full compliance with Sender-ID on outbound messages. If enabled and the check returns a "Pass", message delivery continues. If instead returns a "TempError", the message is retried. Otherwise, the message bounces.

To use this feature, you must currently configure a specific `smtp-source-host` (either at the global or VirtualMTA level).

```
check-mfrom-inbound
```
Scope: source

Type: boolean

Attributes: optional

Default: false

Specifies whether to perform an SPF/mfrom check on in bound messages. If the message has the mfrom and it passes, PowerMTA will add an Authentication-Results header stating the passing results.

44

`check-mfrom-inbound-best-guess`

Scope: source

Type: text

Attributes: optional

Default:

Specifies a dummy SPF record to use for mfrom checking in case a domain being checked does not publish a SPF record.

`check-iprev-inbound`

Scope: source

Type: boolean

Attributes: optional

Default: false

Determines if PowerMTA performs and inbound DNS PTR check on the IP and EHLO hostname used for the connection.

`trace-iprev-check`

Scope: source

Type: boolean

Attributes: optional

Default: false

If enabled, preforms a DNS resolution trace of the lookup for debugging purposes.

`reject-iprev-check-temperror`

Scope: source

Type: boolean

Attributes: optional

Default: false

If the DNS PTR check returns a temp error, instructs PowermTA to reject or accept the message.

reject-iprev-check-permerror

      Scope: source

      Type: boolean

      Attributes: optional

      Default: false

      If the DNS PTR check returns a permanent error, instructs PowerMTA to reject or accept the message.

reject-iprev-check-fail

      Scope: source

      Type: boolean

      Attributes: optional

      Default: false

      If the DNS PTR check returns a failure, instructs PowerMTA to reject or accept the message.

suppress-local-dsn

      Scope: source

      Type: `boolean`

      Attributes: optional

      Default: `false`

      Disables the generation of DSNs for invalid recipients (whether they were suppressed, or invalid for any other reason) that were accepted into the spool.

dsn-return-default

      Scope: source

      Type: `full`, `headers` or `system`

      Attributes: optional

      Default: `system`

      Specifies the default for the DSN `RET` parameter, i.e., whether the full message body or only its headers should be returned in a DSN delivery report. `system` specifies that the system default should be used. This directive is overridden by the use of "dsn-format plain-text".

dsn-format

      Scope: domain

      Type: {standard|plain-text}

      Attributes: optional

      Default: `standard`

      The default is "standard", which has PowerMTA send a DSN in the standard format. If set to "plain-text", the "message/delivery-status" portion of the DSN report is delivered instead of the full report. The portion is delivered within a mime-type of "text/plain", thus allowing this data to pass through (overzealous) email firewalls that strip all "attachments"

custom-dsn-from-header

      Scope: Global

      Type: Email

      Attributes: optional

      Default: `MAIL FROM`

Used to configure the content of 'From' header in DSN reports

default-virtual-mta

> **Scope:** source
>
> **Type:** VirtualMTA name
>
> **Attributes:** optional
>
> **Default:** none

Specifies the default VirtualMTA (or VirtualMTA pool) to select for all messages received from the source. If `process-x-virtual-mta` is also enabled for the source, the default VirtualMTA can be overridden by means of a `x-virtual-mta` header.

"`by-smtp-source-ip`" can be used in place of a VirtualMTA name. When configured in this way, on an inbound SMTP connection PowerMTA will use as the default VirtualMTA whose "smtp-source-host" setting equals the destination IP address on the inbound connection.

Example:

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 vmta1.port25.com
</virtual-mta>

<source 0/0>
    default-virtual-mta by-smtp-source-ip
</source>
```

In the above example, connecting to IP address 1.2.3.4 would automatically route the message through VirtualMTA "mta1". Also, during the connection from the external mailer to PowerMTA, PowerMTA would respond with the host name configured in the VirtualMTA.

reject-invalid-virtual-mta

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** true

Deprecated.

Specifies that PowerMTA should reject messages when an invalid or null VirtualMTA is selected.

48

disconnect-on-transient-error

    Scope: source

    Type: boolean

    Attributes: optional

    Default: false

Specifies that PowerMTA should close connections instead of sending transient SMTP errors. This is useful in situations where the feeding software cannot process 4xx errors.

hide-message-source

    Scope: source

    Type: boolean

    Attributes: optional

    Default: false

Specifies that PowerMTA should attempt to hide the source of the message while delivering e-mail from this source. Currently, this just means that the name and IP address of the MTA from which PowerMTA received this message will not be included in the `Received:` header added. Hiding the message's source may be desirable, for example, for security purposes, to avoid revealing details from the internal network from which the message was submitted.

log-resolution

    Scope: domain

    Type: boolean

    Attributes: optional

    Default: false

Specifies that PowerMTA should log information on DNS (name and routing) lookups performed for the domain. This information is generally useful when debugging DNS-based connectivity problems (such as misconfigured DNS entries).

Example:

```
<domain hotmail.com>
    log-resolution true
</domain>
```

log-transfer-failures

    Scope: domain

    Type: boolean

    Attributes: optional

    Default: false

Specifies whether PowerMTA should write a warning to the logging file when connections fail while transferring message's contents (headers and body). Setting this directive to `true` may help when diagnosing delivery problems, especially duplicate or partial deliveries.

max-message-size

    Scope: source

    Type: number

    Attributes: optional

    Default: unlimited

Specifies the maximum size allowed in messages received through SMTP. If set to unlimited, no explicit limit is enforced.

max-message-hops

    Scope: source

    Type: number

    Attributes: optional

    Default: 100

Specifies the maximum number of "Received" headers to accept in a message.

max-rcpt-per-message

    Scope: source,domain

    Type: number

    Attributes: optional

    Default: 0

Specifies the maximum number of recipients accepted in each message. If set to zero, no explicit limit is enforced.

too-many-rcpts-fails-message

> **Scope:**   source
>
> **Type:**   boolean
>
> **Attributes:** optional
>
> **Default:**   false

> Specifies how PowerMTA should enforce the `max-rcpt-per-message` limit. If enabled and too many recipients are passed for a message, rather than reject the extraneous recipient(s) (`RCPT` SMTP command), PowerMTA rejects the entire message (`DATA` command).

suppression-lists

> **Scope:**   source
>
> **Type:**   suppression list name
>
> **Attributes:** optional
>
> **Default:**   none

> Specifies the suppression-list(s) to be used for suppressing addresses. Multiple lists may be specified, separating each list name with a comma. See section 10.14 for more information.

pattern-list

> **Scope:**   source
>
> **Type:**   pattern list name
>
> **Attributes:** optional
>
> **Default:**   none

> Specifies that messages received from the source are to be matched against the given pattern list. The pattern list referenced must precede the source definition in the configuration file. Section 3.2.16 describes pattern lists in greater detail.

process-x-envid

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should process `x-envid` headers. If set to `true` and one such header is included, PowerMTA will set the message's (DSN) envelope ID to the header's body, as well as remove the header from the message. If set to `false`, PowerMTA will ignore this header, leaving it in the message if present.

While the envelope ID can be set by using the `ENVID` parameter in the SMTP `MAIL` command, this directive is useful when the software you use to submit the messages does not allow you to control it.

process-x-dkim-options

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should process `x-dkim` headers. If set to `true` and one such header is included, PowerMTA will use any values in the header for signing messages with DKIM if DKIM signing is configured. If set to `false`, PowerMTA will ignore this header, leaving it in the message if present.

jobid-prefix

> **Scope:** source
>
> **Type:** string
>
> **Attributes:** optional
>
> **Default:** none

Allows one to have PowerMTA add a custom prefix to the data supplied in the x-job: header in the message. A special reserved {auth} prefix has PowerMTA adding the actual authenticated user as the prefix, allowing for example, a relay service to more easily track messages sent through by PowerMTA by each authenticated user.

```
<source 10.5.5.10>
  job-prefix customer1
</source>
```

process-x-job

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should process `x-job` headers. If set to `true` and one such header is included, PowerMTA will set the job ID for the message to the job ID given in the `x-job` header. The job ID must not include any non-printable or white space characters. A message is supposed to include at most one `x-job` header. If using Mailmerge, you must use the *jobid variable, and not the x-job header.

```
x-job: abc123
```

jobid-header

> **Scope:** source
>
> **Type:** string
>
> **Attributes:** optional
>
> **Default:**

Allows setting an alternate header for which the message's jobId is set in place of using x-job. If other ways to set a jobId (such as x-job processing, if enabled, or *jobId mail merge variable) apply, they take precedence.

retain-x-job

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should retain a processed `x-job` header in the message. If set to `true`, PowerMTA will keep the `x-job` header in the message when sending it on, otherwise that header will be removed. This option is only used if `process-x-job` is set to `true` and the message contains an `x-job` header.

53

process-x-virtual-mta

> **Scope:** source
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

Specifies whether PowerMTA should process `x-virtual-mta` headers. If set to `true` and one such header is included, PowerMTA will select the VirtualMTA specified in the header's body for the message, as well as remove the header from the message. If set to `false`, PowerMTA will ignore this header, leaving it in the message if present.

You should enable this directive for the source IP addresses from which you feed messages if you would like to use the `x-virtual-mta` header to select which VirtualMTA to use. You should probably not enable this globally (e.g., in the `0/0` entry) since this way any messages coming into PowerMTA would be scanned for that header. That would apply, for example, even for remote bounces being received through PowerMTA.

retain-x-virtual-mta

> **Scope:** source
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

Specifies whether PowerMTA should retain a processed `x-virtual-mta` header in the message. If set to `true`, PowerMTA will keep the `x-virtual-mta` header in the message when sending it on, otherwise that header will be removed. This option is only used if `process-x-virtual-mta` is set to `true` and the message contains an `x-virtual-mta` header.

smtp-await-slot

> **Scope:** global
> **Type:** boolean
> **Attributes:** optional
> **Default:** false

Specifies whether PowerMTA should wait for a new connection slot when a new (incoming) SMTP connection request is received but no more slots are available. If set to false, PowerMTA responds with a `421` greeting, indicating that the service is not available, and closes the connection.

source-group

> **Scope:**   source
>
> **Type:**   string
>
> **Attributes:** optional
>
> **Default:**

Specifies the name of a source group to associate the given source to, allowing linking sources to groups (and thus grouping sources). The directive affects incoming SMTP connections from the given group of `<source>` tags. The directive allows for specifying certain source configuration items whose functionality require them to be grouped by name. The shared resources will be defined in the `<source-group>` tag. To add a `<source>` to the shared resource, specify the name of the `<source-group>` in the `<source>` tag. An example would look similar to the following:

```
<source-group localSources>
    max-smtp-in 10
    reserved-smtp-in 5
</source-group>

<source 192.168.0.30>
    smtp-service yes
    always-allow-relaying yes
    source-group localSources
</source>

<source 192.168.0.40>
    smtp-service yes
    always-allow-relaying yes
    source-group localSources
</source>
```

Each `<source>` that defines the source-group directive of the same name will be controlled by the settings in the named `<source-group>` tag.

max-smtp-in

> **Scope:**   source-group
>
> **Type:**   number
>
> **Attributes:** optional
>
> **Default:**   unlimited

Specifies a limit for the number of incoming connections from that source group.

reserved-smtp-in

> **Scope:** source-group
> **Type:** number
> **Attributes:** optional
> **Default:** unlimited

Specifies that a certain number of inbound connections are to be reserved to connections from the given source group.

smtp-ip

> **Scope:** global
> **Type:** IP address
> **Attributes:** optional
> **Default:** all local IP addresses

On a multi-homed host, this directive specifies the IP address on which PowerMTA is to listen for incoming connections. Doing so frees up the SMTP port for other uses on all other IP addresses — as long as the other SMTP software can be instructed to bind itself to specific IP addresses. In a different scenario, for security reasons you may also want to have PowerMTA listen for connections on an internal, non-routable IP address only.

See also `smtp-listener` below.

Example:

```
# listen on internal IP address only
smtp-ip 10.0.0.1
```

smtp-port

> **Scope:** global
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** 25 (standard SMTP port)

Specifies the TCP port number to use for listening for incoming SMTP connections, allowing you to "run PowerMTA on a non-standard port". This can be useful if a second mailer handles all incoming traffic while PowerMTA is fed through another TCP port. If smtp-port is set to 0, PowerMTA does not listen for incoming connections (and feeding is only possible through the submission APIs).

See also smtp-listener below.

Example:

```
# use port 2525 since port 25 is used by mailer X
smtp-port 2525
```

smtp-data-timeout

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** 10m

Specifies the amount of time that PowerMTA will wait for data from the remote end during data reception phase (after DATA/BDAT command) of the message. Applies to inbound connections only.

smtp-command-timeout

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** 10m

Specifies the amount of time that PowerMTA will wait for data to be sent from a remote end during inbound SMTP command exchanges.

process-x-schedule

      Scope: source

      Type: boolean

      Attributes: optional

      Default: false

Specifies whether PowerMTA should process the x-schedule header if included in the message. If set to true and one such header is included, PowerMTA will respect the schedule defined in the header, and only attempt to deliver messages per the schedule. If set to false, PowerMTA will ignore this header and attempt to deliver normally, without any time based schedule.

retain-x-schedule

      Scope: source

      Type: boolean

      Attributes: optional

      Default: false

Specifies whether PowerMTA should leave the x-schedule header in the message when messages are delivered (when using PowerMTA's Schedule Delivery Control feature). While this option is generally used in conjunction with the process x-schedule header option, they are completely independent of each other. Both can be set to false for example, which has PowerMTA ignoring the header with regards to processing however still removing the header before delivery.

retain-x-dkim-options

      Scope: source

      Type: boolean

      Attributes: optional

      Default: false

Specifies whether PowerMTA should retain the x-dkim-options header when sending a message to a remote mail server.

```
smtp-listener
```
        Scope: global

        Type: *IP address:port*

        Attributes: optional

        Default: none

Specifies an IP address and port to listen for incoming SMTP connections. Multiple `smtp-listener` entries can be specified in a configuration file, each of which specifying a different IP address and/or port. 0.0.0.0 may be used more than once to set multiple ports for all IP addresses on the system. A CIDR IP address range can also be specified, however if this is used, the address 0.0.0.0 cannot be used. If any `smtp-listener` directives are specified, PowerMTA ignores the `smtp-ip` and `smtp-port` directives. However, if no `smtp-listener` directives are specified, one listener is implicity started with the address given by `smtp-ip` and `smtp-port`.

`smtp-listener` accepts an optional `source=...` parameter which allows you to specify a `<source>` entry with settings to override those obtained from the normal matching by source IP address.

Example:

```
smtp-listener 1.2.3.4:25 source=mta1
smtp-listener 0.0.0.0:26

<source mta1>
    default-virtual-mta mta1
</source>

<source 10.0.0.0/8>
    allow-mailmerge yes
    always-allow-relaying yes
</source>
<source 0/0>
    allow-mailmerge no
    always-allow-relaying no
</source>
```

In the example above, VirtualMTA `mta1` is selected by default for any messages received through `1.2.3.4`. Permission for relaying and for using the mailmerge extensions are granted to connections coming from 10.*, whether they connect to 1.2.3.4 or not.

Warning If you specify `always-allow-relaying` on a `source` selected from a `smtp-listener` directive, anybody who can connect to the given IP address will be allowed to relay. In doing so, you might be creating an "open SMTP relay".

smtp-service

>**Scope:** source
>
>**Type:** boolean
>
>**Attributes:** optional
>
>**Default:** true

Specifies whether to allow access to the SMTP service to connections from the given source. If set to `false`, PowerMTA issues a 5XX SMTP greeting like

```
521 hostname does not accept mail from you
```

instead of the regular one:

```
220 hostname (PowerMTA version) ESMTP service ready
```

Note that since the default is to grant SMTP service, if you only wish to restrict service to specific sources you must specify that in the `0/0` entry, as in the example below.

Example:

```
<source 10.0.0.1>        # we feed from 10.0.0.1
    smtp-service yes
    always-allow-relaying yes   # normally desirable for feeder sources
</source>
<source 0/0>
    smtp-service no
</source>
```

allow-smtputf8

>**Scope:** source
>
>**Type:** boolean
>
>**Attributes:** optional
>
>**Default:** true

Specifies whether one can submit internationalized email addresses per RFC6531 to PowerMTA from the connecting IP address. When set to yes (or true), PowerMTA will list "SMTPUTF8" in the list of extended SMTP commands supported for the connecting IP, allowing the submitter to use internationalized email addresses in the SMTP envelope. Note that if the remote gateway for the domain does not advertize support for SMTPUTF8, PowerMTA will subsequently bounce the message(s) for the recipient out of the queue.

3.3.5 Web-Based Monitor Directives

`http-mgmt-port`

Scope: global

Type: number

Attributes: optional

Default: 8080

Specifies the TCP port number for the web-based monitor. Since there is no standard port for this application, other software running on the same host may already be using that port. In this case, change the port to some free port number. If `http-mgmt-port` is set to 0, no web-based monitor is started.

Example:

```
# use port 8888 since our proxy uses 8080
http-mgmt-port 8888
```

Changing this directive requires a restart of the pmtahttp service.

```
http-access
```
Scope: global
Type: IP value
Attributes: Required
Default: 127.0.0.1 monitor

The new directive has the format "http-access CIDR {none|monitor|operator|admin}", where CIDR is an IP address or CIDR mask specification. The keyword after the CIDR gives the level of access: none, monitor (read-only), operator (monitor level plus view configuration and run commands) or admin (operator level plus edit config). Admin access is intended for things that actually modify PowerMTA's state (configuration, LAK, queues, etc.). The directive can be specified several times, and builds an access list in which IP addresses are matched from top to bottom.

Example:
```
# allow internal hosts access to the web-based monitor
http-access 10.0.0.0/8 monitor
```

Use of this directive requires a restart of the pmtahttp service. Please see the pmtahttp.log file for errors relating to use of this directive. The pmtahttp.log file should indicate the IP address of any attempted accesses. For quick testing the directive may be set to allow all IPs with the following:

Example:
```
# allow all IPs to the web-based monitor for testing only
http-access 0/0 monitor
```

Do not leave this set for purposes other than testing as this will allow anyone in the world to connect to the web monitor of PowerMTA.

```
http-mgmt-source & no-http-mgmt-source
```
Scope: global
Type: boolean
Attributes: Required
Default:

These functions have been deprecated. Please use http-access.

Example:
```
# allow  internal  hosts  access  to  the  web-based
monitor
http-access 10.0.0.0/8 monitor
```

3.3.6 Logging Directives

`http-log-data`

> **Scope:**   global
> **Type:**   boolean
> **Attributes:** optional
> **Default:**   false

> Specifies whether web monitor data exchanges are logged or not. More verbose than http-log-requests.

`http-log-requests`

> **Scope:**   global
> **Type:**   boolean
> **Attributes:** optional
> **Default:**   false

> Specifies whether web monitor data exchanges are logged or not.

`log-auto-rotation`

> **Scope:**   global
> **Type:**   boolean
> **Attributes:** optional
> **Default:**   true

> Specifies whether the logging files are to be rotated automatically at midnight. See the `log-rotate` directive for more information on file rotation.

63

```
log-file
```

Scope: global

Type: file name

Attributes: required

Default:

Specifies the name of PowerMTA's logging file. On Windows, it can be either specified with a path name relative to the installation directory or with an absolute path. On Unix it must always be specified as an absolute path. The log-file directive controls the location of the PowerMTA log file, the HTTP log file, and the SNMP log file. There is no method to control their location independently.

Example:

```
#windows
log-file log\pmta.log

#linux
log-file /var/log/pmta/pmta.log
```

```
log-rotate
```

Scope: global

Type: number

Attributes: optional

Default: 8

Specifies the number of files to keep when rotating the logging files. File rotation is a practical way of preventing files from growing too large and, at the same time, ensure that the most recent data is available in case it is needed.

When the log files are rotated, PowerMTA renames the existing logging files, attaching a number to them, starting with 1 for the most recent file. On Unix, assuming a file named `log`, that's `log.`N` -> log.`$N+1$, `log -> log.1` and on Windows (assuming `log.txt`), `log-`N`.txt -> log-`$N+1$`.txt`, `log.txt -> log-1.txt`. It then creates a new file (either `log` or `log.txt`) to which it continues writing logging information.

The number passed includes the file currently being written to, so a value of 1 means only the current file is kept (but no older files), 2 means the current file plus one old file, and so on. A value of 0 disables rotation entirely.

To trigger manual rotation run the following command:

```
pmta rotate log
```

`log-resolution`

>**Scope:** domain
>**Type:** boolean
>**Attributes:** optional
>**Default:** false

>Specifies that PowerMTA should log information on DNS (name and routing) lookups performed for the domain. This information is generally useful when debugging DNS-based connectivity problems (such as misconfigured DNS entries).

>**Example:**

```
<domain hotmail.com>
    log-resolution true
</domain>
```

`log-connections`

>**Scope:** domain, source
>**Type:** boolean
>**Attributes:** optional
>**Default:** false

>Instructs PowerMTA to write an entry in the log at the beginning and end of each connection.

>**Examples:**

```
<domain hotmail.com>
    log-connections yes
</domain>

# log *all* incoming connections
<source 0/0>
    log-connections yes
</source>
```

```
log-commands
```
       Scope: domain, source

       Type: boolean

       Attributes: optional

       Default: false

Instructs PowerMTA to log the full SMTP protocol exchanges between itself and the receiving or sending mailer. Since it displays at which point of the delivery process an error occurs, enabling this directive is generally the first and most useful step in debugging connectivity problems.

Example:

```
<domain aol.com>
    log-commands yes
</domain>
```

Log output:

```
2003-06-29 15:32:00 ( 20)starting aol.com
2003-06-29 15:32:00 ( 20)connecting to yh.mx.aol.com (205.188.157.1)
2003-06-29 15:32:00 ( 20)>>> 220-rly-yh01.mx.aol.com ESMTP relay_in.5; Thu, 29 Jun 2003
09:27:58 -0400
2003-06-29 15:32:00 ( 20)>>> 220-America Online (AOL) and its affiliated companies do
not
2003-06-29 15:32:00 ( 20)>>> 220-     authorize the use of its proprietary computers and
computer
2003-06-29 15:32:00 ( 20)>>> 220-     networks to accept, transmit, or distribute
unsolicited bulk
2003-06-29 15:32:00 ( 20)>>> 220     e-mail sent from the internet.
2003-06-29 15:32:00 ( 20)<<< EHLO hazmat.port25.com
2003-06-29 15:32:00 ( 20)>>> 250-rly-yh01.mx.aol.com hazmat.port25.com
2003-06-29 15:32:00 ( 20)>>> 250 HELP
...
```

```
log-data
```
Scope: domain, source

Type: boolean

Attributes: optional

Default: false

This directive is useful for debugging protocol or interoperability problems between the mailers, for it logs every byte sent, both in ASCII and hexadecimal. This would help show things like whether your messages may be finishing its lines with just LF (line feeds) instead of the CRLF (carriage return, line feed) pair as prescribed by the standards, and the hex data allows you to identify non-printable characters (such as CR and LF) based on their hex codes.

Example:

```
<domain yahoo.com>
    log-data yes
</domain>
```

Log output:

```
2003-06-29 09:53:59 ( 19)starting yahoo.com
2003-06-29 09:53:59 ( 19)connecting to mx1.mail.yahoo.com (128.11.68.155)
2003-06-29 09:54:04 ( 19)>>> rd 53
2003-06-29 09:54:04 ( 19)3232302059536D7470206D7461313334 220 YSmtp mta134
2003-06-29 09:54:04 ( 19)2E6D61696C2E7961686F6F2E636F6D20 .mail.yahoo.com
2003-06-29 09:54:04 ( 19)45534D5450207365727669636520726  ESMTP service re
2003-06-29 09:54:04 ( 19)6164790D0A                       ady..
2003-06-29 09:54:04 ( 19)<<< wr 24
2003-06-29 09:54:04 ( 19)45484C4F2068617A6D61742E706F7274 EHLO hazmat.port
2003-06-29 09:54:04 ( 19)32352E636F6D0D0A                 25.com..
2003-06-29 09:54:04 ( 19)>>> rd 75
2003-06-29 09:54:04 ( 19)3235302D6D74613133342E6D61696C2E 250-mta134.mail.
2003-06-29 09:54:04 ( 19)7961686F6F2E636F6D0D0A3235302D38 yahoo.com..250-8
2003-06-29 09:54:04 ( 19)4249544D494D450D0A3235302D53495A BITMIME..250-SIZ
2003-06-29 09:54:04 ( 19)4520333134353732380D0A3235302050 E 3145728..250 P
2003-06-29 09:54:04 ( 19)4950454C494E494E470D0A            IPELINING..
2003-06-29 09:54:04 ( 19)<<< wr 43
2003-06-29 09:54:04 ( 19)4D41494C20465243F4D3A3C696E666F40 MAIL FROM:<info@
2003-06-29 09:54:04 ( 19)706F727432352E636F6D3E20424F4459 port25.com> BODY
2003-06-29 09:54:04 ( 19)3D384249544D494D450D0A            =8BITMIME..
2003-06-29 09:54:04 ( 19)>>> rd 33
2003-06-29 09:54:04 ( 19)3235302073656E646572203C696E666F 250 sender <info
2003-06-29 09:54:04 ( 19)40706F727432352E636F6D3E206F6B0D @port25.com> ok.
2003-06-29 09:54:04 ( 19)0A                                .
2003-06-29 09:54:04 ( 19)<<< wr 29
...
```

```
log-transfer-failures
```
>   **Scope:**   domain
>   **Type:**   boolean
>   **Attributes:** optional
>   **Default:**   false

Specifies whether PowerMTA should write a warning to the logging file when connections fail while transferring message's contents (headers and body). Setting this directive to `true` may help when diagnosing delivery problems, especially duplicate or partial deliveries.

```
log-file-world-readable
```
>   **Scope:**   global
>   **Type:**   boolean
>   **Attributes:** optional
>   **Default:**   false

Grants the web monitor access to read and display the log files that have been rotated.

```
log-disabled-ips
```
>   **Scope:**   domain
>   **Type:**   boolean
>   **Attributes:** optional
>   **Default:**   false

Turns on logging to show when and which IPs were disabled.

```
max-events-recorded
```

Scope: domain

Type: number

Attributes: optional

Default: 10

Specifies the number of *events* PowerMTA is to record about the given domain. These events are not written to the logging file, but just recorded in memory. They can be viewed using the `pmta show domains` and the `pmta show topdomains` commands, or via the web interface. While recording such events can be very useful when investigating delivery problems, may have have an impact on memory consumption. Currently only error events are recorded.

Example:

Configuration file:

```
<domain port25.com>
    max-events-recorded 3
</domain>
```

At the command prompt:

```
$ pmta show dom port25.com --errors
---------------------------domain --#rcpt ---kbytes conn last error----------
                        port25.com       4       0.5    0 ETIMEDOUT connect...

  2003-06-19 16:40:10 ETIMEDOUT connecting to mail.port25.com (193.96.192.241)
  2003-06-19 16:40:10 ETIMEDOUT connecting to mail.port25.com (193.96.192.241)
  2003-06-19 16:40:10 ETIMEDOUT connecting to mail.port25.com (193.96.192.241)

1 of 10 domains shown.
```

3.3.7 Accounting Directives

The accounting File can be written to CSV or pipe. For more information on the accounting file, see Section 11.

disable-acct-records

> **Scope:** Domain
>
> **Type:** {d,b,t,tq}
>
> **Attributes:** Optional
>
> **Default:** None

> Allows one to disable the creation of one or more than one type of accounting file record on a per queue (domain / virtualMTA) basis. For example, there are times when a user wants PowerMTA to discard certain messages without creating an accounting file record for these messages when discarded.   By default, PowerMTA would create a "d" (delivered) record for this, since it was delivered to discard, however the creation of a delivered record can be disabled with this option.

delete-after

> **Scope:** acct-file
>
> **Type:** {#d|never}
>
> **Attributes:** optional
>
> **Default:** 8d (8 days)

> This directive tells PowerMTA how long to keep the accounting file before deleting it from the server. "delete-after" only applies if "move-to" isn't active" (and move-to must point to a different location than the directory where the file was originally created).

move-to

> **Scope:** acct-file
>
> **Type:** directory
>
> **Attributes:** optional
>
> **Default:** files not moved

> This directive tells PowerMTA where to move the accounting file when using `move-interval` or `max-size`. If move-to isn't set, no moving is done (move-interval is silently ignored).   Directory must be on the same volume as the accounting files. Moving across a network is not supported.

move-interval

> **Scope:** acct-file
> **Type:** time interval
> **Attributes:** optional
> **Default:** 5m

This directive tells PowerMTA at what time interval to move the accounting file and start a new one.

max-size

> **Scope:** acct-file
> **Type:** {n{B|K|M|G|T}
> **Attributes:** optional
> **Default:** 50M

The max size of the accounting file in Bytes, Kilobytes, Megabytes, Gigabytes, or Terabytes before PowerMTA moves the accounting file and starts a new one. Minimum size is 1M (1000000B & 1000K) & Maximum size is 1T. This directive applies both with and without "move-to". If "move-to" is not specified, a new file is started in place (with a higher –NNNN in the file name). The max value is 8388607T.

retry-interval

> **Scope:** acct-file
> **Type:** time interval
> **Attributes:** optional
> **Default:** 5m

Retries the pipe program in a regular interval in case of a failure. This should only be used when using pipe to write an accounting file.

iso-times

> **Scope:** acct-file
> **Type:** boolean
> **Attributes:** optional
> **Default:** yes

Allows requesting that PowerMTA formats time stamps in the accounting file (or pipe) in ISO format.

`user-string`

Scope: acct-file

Type: string

Attributes: optional

Default:

Specifies a custom string to be stored along with other MTA information in theaccounting file. The field "userstring" also needs to be added to the record-fields directive for this to show up in the accounting file.

`write-timeout`

Scope: acct-file

Type: time interval

Attributes: optional

Default: 1m

Allows for setting write timeouts in accounting pipes, so that an unresponsive accounting application doesn't hold up PowerMTA indefinitely. Only used when pipeing accounting data to another file.

`map-header-to-field`

Scope: acct-file

Type: string

Attributes: optional

Default:

Allows for mapping a header to a specific field in the accounting file. This is useful if for example the from header needs to be mapped to the SMTP MAIL FROM address ("orig" field in the accounting file). An example would be like the following:

```
map-header-to-field b header_x-foo rcpt
map-header-to-field d header_x-bar rcpt
map-header-to-field d header_x-baz rcpt
map-header-to-field d header_x-foo orig
```

In the above example header_x-foo gets mapped to the rcpt field in bounced records, and header_x-bar gets mapped to the rcpt field for delivered records. Multiple entries may be specified to map multiple headers to the same field.

The tq record type cannot be used.

The order of the directives is significant when using multiple ones for the same record type and field. In the above example the rcpt field will contain the content of the x-foo header if x-foo exists, if not it will contain the content of the x-bar header if that one exists. Otherwise the content of the x-baz header. If that one also doesn't exist it will contain the original value of the orig field (i.e. the rcpt field will not be overwritten).

The same header can be mapped to multiple fields (e.g. header_x-foo above) and will be written to all fields mapped.

`records`

> **Scope:**   acct-file
> **Type:**   one or more comma-separated record types
> **Attributes:** optional
> **Default:**   d,b

This directive tells PowerMTA which record types to include in the accounting file. There are 8 record types: d, b, t, tq, r, f, rb, & rs. It may only be defined once per <acct-file> with multiple types separated by commas.

- "d" is for delivered records and will include all messages that were successfully delivered. "delivery" can be used in place of "d".

- "b" is for bounced records and will include all locally generated bounces (synchronous bounces). Note that bounces generated by remote mail servers (asynchronous bounces) will not be included. "bounce" can be used in place of "b".

- "t" is for transient recipient level errors only (such as mailbox full). "transient" can be used in place of "t".

- "tq" is for queue wide transient errors and will include information about delivery attempts for a given domain/vmta. "transient-queue" can be used in place of "t".

- "r" is for inbound records and will include information about messages submitted to PowerMTA for delivery. "receipt" can be used in place of "r".

- "f" is for feedback loop emails if PowerMTA is configured to look for feedback loop emails with the <feedback-loop-processor>. "feedback-loop" may be used in place of "f". See Section 12.7 for more information.

- "rb" is for remote bounce emails if PowerMTA is configured to look for remote bounce emails with the <bounce-processor>. "remote-bounce" may be used in place of "rb". See Section 12.6 for more information.

- "rs" is for remote status emails if PowerMTA is configured to look for remote status emails with the <bounce-processor>. "remote-status" may be used in place of "rs". See Section 12.6 for more information.

Due to the fact that there can be many delivery attempts for one message, it is suggested that type "t" and/or "tq" should only be used when needed (debugging, troubleshooting, etc.) as it may cause large file sizes. It is also recommended when using this setting to have the "t" or "tq" type records written to their own file, as so not to cause unwanted data in the primary accounting file.

`record-fields`

> **Scope:** acct-file
>
> **Type:** record type followed by a comma-separated list of field names
>
> **Attributes:** optional
>
> **Default:** *

This directive specifies the fields to be included in the records defined via the "record" directive, in the accounting file. By default all per record type are included. To include only a subset of all the fields, simply add the desired fields in a comma separated list. To include a custom header, add the record field header_XXX, where XXX is the name of the custom field you wish to log. For example, to log all fields and a custom header, the directive would look like this:

```
record-fields delivery *,header_Message-Id
```

To include only a subset of the fields, list the required fields similar to the following:

```
record-fields bounce timeLogged,orig,rcpt,dlvSourceIp,vmta
```

To exclude fields from the default, you can use an "!" to exclude the unwanted fields. For example, to include all fields except orcpt you would use:

```
record-fields d *,!orcpt
record-fields b *,!orcpt
```

`world-readable`

> **Scope:** acct-file
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** no

This directive tells PowerMTA what read permissions to set on the accounting file.

75

`count-moved-records`

Scope: acct-file

Type: boolean

Attributes: optional

Default: false

This directive tells PowerMTA whether or not to count the records in CSV files moved. The record counts are written to the logging file along with the message that indicates that the file was moved.

`sync`

Scope: acct-file

Type: boolean

Attributes: optional

Default: no

Determines whether the OS' buffers for the accounting file are flushed regularly. This helps ensure that the accounting data isn't corrupted in case of a system crash.

`deliver-local-dsn`

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** true

This directive instructs PowerMTA whether or not to generate and deliver a Delivery Status Notification (DSN) for a message. DSNs are more widely referred to or known as bounce reports, however non-bounce DSNs exist as well.

Regardless if set to true or false, PowerMTA will write the final bounce detail in the PowerMTA accounting file when PowerMTA receives a permanent/5xx error from the remote host during the connection, or when the error is generated by PowerMTA. The default setting is true, which instructs PowerMTA to generate and attempt to deliver the DSN report, as well as write the bounce detail in the accounting file.

If set to false, PowerMTA does not even create the DSN report, it just efficiently writes the bounce detail in the accounting file for processing. A setting of false is the correct setting for sites that parse only the accounting file data for the locally generated bounce reports (also referred to "sychronous" or "inline" bounces), and do not want to have any bounce reports generated/sent to the SMTP MAIL FROM address.

For consistency with previous versions of PowerMTA, these records are still labeled as <repDlv> records in the accounting file. Since the bounce detail is now logged when the error is received and not when the bounce report is actually delivered, <repDlv> records will not contain certain fields that existed previously, that related to the actual delivery of the bounce report. The fields that will not be included are: <dlvFrom>, <dlvThrough>, and <size>, since these were all determined at the time the bounce report was to be delivered.

The deliver-local-dsn directive is as per domain directive, and which should be defined for the SMTP MAIL FROM domain, since this is the address that DSN reports are sent. To have it apply to all local bounce reports regardless of the SMTP MAIL FROM domain (e.g. if your messages have a different SMTP MAIL FROM domain depending on the customer/client or campaign) simply define this directive within the global <domain *> definition. For example, the following setting would apply to messages whose SMTP MAIL FROM domain was bounce.port25.com, regardless of the original recipient address:

```
<domain bounce.port25.com>
    deliver-local-dsn   no
</domain>
```

77

3.3.8 General Queueing and Delivery Directives

`bounce-after`

> **Scope:**   domain
> **Type:**   time interval
> **Attributes:** optional
> **Default:**   4 days, 12 hours

> Determines the maximum amount of time PowerMTA will continue to try to deliver messages to the specified destination domain. Messages older than this time interval are bounced, i.e., a (non-)delivery report in DSN format will be sent back to the originator.

`bcc-upon-delivery`

> **Scope:**   domain
> **Type:**   String
> **Attributes:** optional
> **Default:**   none

> Automatically BCC emails delivered to specific domains with the given address. A null string with "" may be used to override an inherited setting.

`allow-priority-interruption`

> **Scope:**   domain
> **Type:**   boolean
> **Attributes:** optional
> **Default:**   yes

> When set to "no", this directive prevents interruptions by higher priority queues.

`allow-priority-interruption-during-transfer`

> **Scope:**   domain
> **Type:**   boolean
> **Attributes:** optional
> **Default:**   no

> When set to 'no', the interruption is delayed until a connection is available. When set to 'yes', the interruption to an existing connection happens immediately. This directive applies to SMTP queues only.

queue-priority

      Scope: domain

      Type: number

      Attributes: optional

      Default: 50

When PowerMTA needs to deliver email for a queue but no connection slots are available, it looks for a queue with lower priority currently connected and interrupts as many connections as needed, starting with the lowest priority connections and working its way up. A queue's priority is set with a number between 0 (lowest) and 100 (highest); 50 by default.

delivery-priority

      Scope: domain

      Type: number

      Attributes: optional

      Default: 50

Deprecated, use queue-priority.

dk-sign

      Scope: domain

      Type: boolean

      Attributes: optional

      Default: false

      Deprecated.

Specifies whether or not PowerMTA should perform Domain Keys signing on messages destined for this domain. Domain keys must be configured for this directive to take effect.

Please see the `domain-key` directive below for more information.

NOTE: In previous versions, the default for this directive was true, and as such, you may need to change this in your configuration file at the time of an upgrade.

dkim-sign

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether or not PowerMTA should perform DKIM signing on messages destined for this domain. Domain keys must be configured for this directive to take effect.

Please see the `domain-key` directive below for more information.

NOTE: In previous versions, the default for this directive was true, and as such, you may need to change this in your configuration file at the time of an upgrade.

Also accepts `sign-if-x-dkim-options-present` as a way to enable DKIM signing only when an X-DKIM-Options header is present in the email. The directive continues to accept 'yes' and 'no' settings with their existing meaning.

dkim-algorithm

> **Scope:** domain
>
> **Type:** {rsa-sha1|rsa-sha256}
>
> **Attributes:** optional
>
> **Default:** rsa-sha1

Specify which DKIM signing algorithm to use with possible values of "rsa-sha1" and "rsa-sha256".

dkim-query-method

> **Scope:** domain
>
> **Type:** String
>
> **Attributes:** optional
>
> **Default:** none

Specifies the value of the "q=" tag in DKIM signatures.

`dkim-headers`

>
> **Scope:** domain
>
> **Type:** string
>
> **Attributes:** optional
>
> **Default:**

Specifies the headers to be used for DKIM signing. The following headers will always be used if included in the message and cannot be removed:

CC	List-Id	Resent-Date
Content-Description	List-Owner	Resent-From
Content-Id	List-Post	Resent-Message-ID
Content-Transfer-Encoding	List-Subscribe	Resent-Sender
Content-Type	List-Unsubscribe	Resent-To
Date	MIME-Version	Sender
From	Message-ID	Subject
In-Reply-To	References	To
List-Archive	Reply-To	
List-Help	Resent-Cc	

RFC 4871 forbids signing the following headers:

Bcc	Keywords	Return-Path
Comments	Received	
DKIM-Signature	Resent-Bcc	

Example usage:

```
dkim-headers foo                    # exact match one header
dkim-headers foo, bar, baz          # exact match many headers
dkim-headers /foo/                  # regular expression
dkim-headers /foo/, /bar/, /baz/    # many regular expressions
dkim-headers foo, /bar/       # exact match and a regular expression
dkim-headers ""               # no new headers signed.
                              # PMTA continues to sign well known
                              # headers as done before
```

`dkim-identity`

> **Scope:** domain
> **Type:** string
> **Attributes:** optional
> **Default:** sender-or-from

Specifies the domain or email address of the signing identity to be used when signing messages with DKIM. (Supported in the DKIM specification only). If defined, and the domain matches, or is a subdomain of the domain defined in the "domain-key" directive, the domain will be used in place of the Sender or From: header domain for signing and for subsequent key verification by the receiving gateways. This may be useful for those who are sending on behalf of another party, and do not have control of the other party's DNS record. A service provider for example, can sign mail using their own private key, use their own email address/domain as the signing identity, and have their own public key in their own DNS record for validation, all while having the From: header domain in the messages sent being the domain of their customer. The "dkim-identity" address will be seen in the i= parameter in the signature, while the d= parameter is set via the domain used in the "domain-key" directive and which has to match or be the parent domain in the dkim-identity directive.

The directive also takes the keyword "strict-from" which requires an exact match for the From: header's domain to find a matching domain key for signing. Any sub-domain or the sender header will not be used.

Note that receiving gateways will normally judge a message based on the reputation of the domain of the signing identity.

Example:

```
domain-key key1,esp123.com,c:\pmta\m.pem
<domain yahoo.com>
   dkim-sign yes
   dkim-identity @esp123.com
#     dkim-identity sender-or-from
#     dkim-identity email@esp123.com
</domain>
```

In the above example, the "From:" or "Sender:" headers can be a different domain then "esp123.com", but it will be the domain "esp123.com" that will be used for the signing, and, the public key in the TXT record in DNS for the domain "key1._domainkey.esp123.com" will be the one checked for verification.

82

`dkim-identity-fallback`

> **Scope:** domain
> **Type:** string
> **Attributes:** optional
> **Default:** ""

Specifies an alternative DKIM identity to use if the primary identity isn't usable (usually because no keys are matched). This allows PowerMTA to use the dkim key for the From or Sender header if available (eg. domain-key directive), but if the key is not available, PowerMTA switches to using the dkim-identity-fallback for signing (i= specified email address in the signature). Requires the use of dkim-identity.

Example:

```
domain-key key1,esp456.com,c:\pmta\m.pem
<domain yahoo.com>
   dkim-sign yes
   dkim-identity sender-or-from
   dkim-identity-fallback @esp456.com
</domain>
```

`dkim-disallow-adding-headers`

> **Scope:** domain
> **Type:** Comma separated list
> **Attributes:** optional
> **Default:** none

Primarily used to minimize DKIM replay attacks, it allows one to specify headers that invalidate the DKIM signature if the headers are added after the DKIM signing.

The directive takes a comma separated list of headers to include an extra time in the signature's "h" tag. Any headers listed in this directive that are also present in the original message will be signed as well.

Example:

```
<domain yahoo.com>
   dkim-disallow-adding-headers from,to,subject
</domain>
```

```
domain-key
```

Scope: global, virtual mta

Type: see text

Attributes: optional

Default: none

Specifies a domain key to use for messages delivered through the given VirtualMTA or for any messages if configured at the global level. You can specify multiple `domain-key` entries. PowerMTA determines the key to use by going through the keys sequentially and picking the first key whose domain either equals or is a parent of the message's *sending domain*. If there are matching domain keys at both the global level and the VirtualMTA level, the VirtualMTA's domain keys take precedence. The sending domain is that of the `Sender` header, or, if `Sender` is not present, of the `From` header. If no keys match the sending domain, the message is not signed.

The wildcard "*" may be used in place of a domain. This will cause all messages to be signed using the sending domain (extracted from the headers).

For more information on how to set up DomainKeys, please see Section 10.4.2.

Example:

```
<virtual-mta mta1>
    ...
    domain-key m,mail.port25.com,c:\pmta\m.pem
    domain-key base,*,c:\pmta\base.pem
</virtual-mta>
```

or

```
domain-key m,mail.port25.com,c:\pmta\m.pem
domain-key base,*,c:\pmta\base.pem
```

84

invalid-virtual-mta-fallback

    Scope: global

    Type: VirtualMTA name

    Attributes: optional

    Default: none

Deprecated.

Specifies a fallback VirtualMTA to select for messages for which an invalid (non-existant) VirtualMTA was selected. By using this directive, you prevent PowerMTA from bouncing messages for which an invalid VirtualMTA was selected. Instead, the messages are delivered from the fallback VirtualMTA.

This directive only works when feeding PowerMTA. It does not work with reroute-to-virtual-mta or backoff-reroute-to-virtual-mta.

allow-empty-x-virtual-mta

    Scope: source

    Type: Boolean

    Attributes: optional

    Default: no

Specifies PowerMTA should allow an empty or null value for the x-virtual-mta header. If set to yes and such a header is passed, the message will be accepted and then queued to the {default} VirtualMTA.

queue-to

    Scope: domain

    Type: string

    Attributes: optional

    Default: none

Allows for grouping domains with the same MX record together. See section 10.7 for more informaiton.

log-tls

    Scope: domain, source

    Type: boolean

    Attributes: optional

    Default: Yes

Enables TLS-specific logging.

```
log-connections
```

Scope: domain, source

Type: boolean

Attributes: optional

Default: false

Instructs PowerMTA to write an entry in the log at the beginning and end of each connection.

Examples:

```
<domain hotmail.com>
    log-connections yes
</domain>

# log *all* incoming connections
<source 0/0>
    log-connections yes
</domain>
```

86

log-commands

>   **Scope:** domain, source
>   **Type:** boolean
>   **Attributes:** optional
>   **Default:** false

Instructs PowerMTA to log the full SMTP protocol exchanges between itself and the receiving or sending mailer. Since it displays at which point of the delivery process an error occurs, enabling this directive is generally the first and most useful step in debugging connectivity problems.

Example:

```
<domain aol.com>
    log-commands yes
</domain>
```

Log output:

```
2003-06-29 15:32:00 ( 20)starting aol.com
2003-06-29 15:32:00 ( 20)connecting to yh.mx.aol.com (205.188.157.1)
2003-06-29 15:32:00 ( 20)>>> 220-rly-yh01.mx.aol.com ESMTP relay_in.5; Thu, 29 Jun 2003
09:27:58 -0400
2003-06-29 15:32:00 ( 20)>>> 220-America Online (AOL) and its affiliated companies do
not
2003-06-29 15:32:00 ( 20)>>> 220-    authorize the use of its proprietary computers and
computer
2003-06-29 15:32:00 ( 20)>>> 220-    networks to accept, transmit, or distribute
unsolicited bulk
2003-06-29 15:32:00 ( 20)>>> 220     e-mail sent from the internet.
2003-06-29 15:32:00 ( 20)<<< EHLO hazmat.port25.com
2003-06-29 15:32:00 ( 20)>>> 250-rly-yh01.mx.aol.com hazmat.port25.com
2003-06-29 15:32:00 ( 20)>>> 250 HELP
...
```

```
log-data
```

      Scope: domain, source

      Type: boolean

      Attributes: optional

      Default: false

This directive is useful for debugging protocol or interoperability problems between the mailers, for it logs every byte sent, both in ASCII and hexadecimal. This would help show things like whether your messages may be finishing its lines with just LF (line feeds) instead of the CRLF (carriage return, line feed) pair as prescribed by the standards, and the hex data allows you to identify non-printable characters (such as CR and LF) based on their hex codes.

Example:

```
<domain yahoo.com>
    log-data yes
</domain>
```

Log output:

```
2003-06-29 09:53:59 ( 19)starting yahoo.com
2003-06-29 09:53:59 ( 19)connecting to mx1.mail.yahoo.com (128.11.68.155)
2003-06-29 09:54:04 ( 19)>>> rd 53
2003-06-29 09:54:04 ( 19)3232302059536D7470206D7461313334 220 YSmtp mta134
2003-06-29 09:54:04 ( 19)2E6D61696C2E7961686F6F2E636F6D20 .mail.yahoo.com
2003-06-29 09:54:04 ( 19)45534D5450207365727669636520726  ESMTP service re
2003-06-29 09:54:04 ( 19)6164790D0A                      ady..
2003-06-29 09:54:04 ( 19)<<< wr 24
2003-06-29 09:54:04 ( 19)45484C4F2068617A6D6174 2E706F7274 EHLO hazmat.port
2003-06-29 09:54:04 ( 19)32352E636F6D0D0A                25.com..
2003-06-29 09:54:04 ( 19)>>> rd 75
2003-06-29 09:54:04 ( 19)3235302D6D74613133342E6D61696C2E 250-mta134.mail.
2003-06-29 09:54:04 ( 19)7961686F6F2E636F6D0D0A3235302D38 yahoo.com..250-8
2003-06-29 09:54:04 ( 19)4249544D494D450D0A3235302D53495A BITMIME..250-SIZ
2003-06-29 09:54:04 ( 19)4520333134353732380D0A3235302050 E 3145728..250 P
2003-06-29 09:54:04 ( 19)4950454C494E494E470D0A           IPELINING..
2003-06-29 09:54:04 ( 19)<<< wr 43
2003-06-29 09:54:04 ( 19)4D41494C2046524F4D3A3C696E666F40 MAIL FROM:<info@
2003-06-29 09:54:04 ( 19)706F727432352E636F6D3E20424F4459 port25.com> BODY
2003-06-29 09:54:04 ( 19)3D384249544D494D450D0A           =8BITMIME..
2003-06-29 09:54:04 ( 19)>>> rd 33
2003-06-29 09:54:04 ( 19)3235302073656E646572203C696E666F 250 sender <info
2003-06-29 09:54:04 ( 19)40706F727432352E636F6D3E206F6B0D @port25.com> ok.
2003-06-29 09:54:04 ( 19)0A                               .
2003-06-29 09:54:04 ( 19)<<< wr 29
...
```

`max-events-recorded`

Scope: domain

Type: number

Attributes: optional

Default: 10

Specifies the number of *events* PowerMTA is to record about the given domain. These events are not written to the logging file, but just recorded in memory. They can be viewed using the `pmta show domains` and the `pmta show topdomains` commands, or via the web interface. While recording such events can be very useful when investigating delivery problems, may have have an impact on memory consumption. Currently only error events are recorded.

Example:

Configuration file:

```
<domain port25.com>
    max-events-recorded 3
</domain>
```

At the command prompt:

```
$ pmta show dom port25.com --errors
---------------------------domain --#rcpt ---kbytes conn last error----------
                      port25.com      4       0.5    0 ETIMEDOUT connect...

  2003-06-19 16:40:10 ETIMEDOUT connecting to mail.port25.com (193.96.192.241)
  2003-06-19 16:40:10 ETIMEDOUT connecting to mail.port25.com (193.96.192.241)
  2003-06-19 16:40:10 ETIMEDOUT connecting to mail.port25.com (193.96.192.241)

1 of 10 domains shown.
```

```
max-smtp-msg-rate
```
Scope: virtual-mta

Type: {unlimited|0|N/{h|hr|m|min|s|sec}

Attributes: optional

Default: unlimited

Allows for entering per-minute and per-second transfer rates directly. It accepts the following values: "unlimited", "N/h" or "N/hr" (for per-hour), "N/m" or "N/min" (for per-minute) and "N/s" or "N/sec" (for per-second).

```
<virtual-mta customer1>
  max-smtp-msg-rate 6/sec
</virtual-mta>
```

Both deliveries and attempted deliveries are counted towards the limit. The following error will be displayed in the web monitor when the rate limit is reached:

"message rate limit reached (based on max-smtp-msg-rate in configuration)"

When using rate limiting, it is best to establish a baseline without rate limiting to ensure that the limits set are having a positive impact on deliverability.

90

`source-ip-max-msg-rate`

     Scope: domain

     Type: {unlimited|0|N/{h|hr|m|min|s|sec}

     Attributes: optional

     Default: unlimited

IP rate limiting allows for controlling the number of attempted recipients on a per-hour, per-minute and per-second basis for each IP address for each domain/VirtualMTA. Primarily used by sites that define multiple IPs in a single VirtualMTA, and that want to limit the attempted delivery rate for each IP address in the VirtualMTA to the respective domains.

```
<domain comcast.net>
  source-ip-max-msg-rate 1.2.3.4 600/h
</domain>
```

`max-msg-rate`

     Scope: domain

     Type: {unlimited|0|N/{h|hr|m|min|s|sec}

     Attributes: optional

     Default: unlimited

Specifies attempted recipients per time interval. Allows for entering per-minute and per-second transfer rates directly. It accepts the following values: "unlimited", "N/h" or "N/hr" (for per-hour), "N/m" or "N/min" (for per-minute) and "N/s" or "N/sec" (for per-second). "max-msg-rate" and "max-msg-per-hour" modify the same internal setting and thus override each other when set.

```
<domain comcast.net>
  max-msg-rate 6/sec
</domain>
```

Both deliveries and attempted deliveries are counted towards the limit. The following error will be displayed in the web monitor when the rate limit is reached:

"message rate limit reached (based on max-msg-rate in configuration)"

When using rate limiting, it is best to establish a baseline without rate limiting to ensure that the limits set are having a positive impact on deliverability.

`backoff-max-msg-rate`

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** unlimited

This directive has the same function as max-msg-rate, and applies instead of max-msg-rate while a queue is in backoff mode.

Both deliveries and attempted deliveries are counted towards the limit. The following error will be displayed in the web monitor when the rate limit is reached:

"message rate limit reached (based on backoff-max-msg-rate in configuration)"

When using rate limiting, it is best to establish a baseline without rate limiting to ensure that the limits set are having a positive impact on deliverability.

`max-msg-per-hour`

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** unlimited

Deprecated, use max-msg-rate.

`backoff-max-msg-per-hour`

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** unlimited

Deprecated, use backoff-max-msg-rate

```
retry-after
```
Scope: domain

Type: time interval

Attributes: optional

Default: 10 minutes

Specifies the retry interval or intervals for the domain/virtualMTA once the domain/virtualMTA is put in retry mode. The directive is a time interval(s), takes a single interval or comma separated list of intervals, with each having a number along with "d" for days, "h" for hours, "m" for minutes, or "s" for seconds, with no spaces between the parameters.

Example:

```
<domain example.com>
    Retry-after 10m,10m,10m,10m,10m,1h,1h,1h,4h,4h,12h
</domain>
```

In the above example, PowerMTA would try every 1om minutes for the first 5 tries, every hour for the next two tries, every 4 hours for the next two tries, and every 12 hours until the message is bounced.

If during any of the retry intervals an email is delivered then the retry-after period is reset and starts again at the first retry interval. The feature is meant to protect against using resources to continually try and email to a dead domain, like example.com.

```
backoff-retry-after
```
Scope: domain

Type: time interval

Attributes: optional

Default: one hour

This directive has the same function as `retry-after`, and applies instead of `retry-after` while a queue is in backoff mode.

`backoff-notify`

> **Scope:** domain
>
> **Type:** e-mail address
>
> **Attributes:** Optional
>
> **Default:** `""` (no notification sent)

Specifies e-mail addresses to notify when any of the queues for the given domain enters or leaves backoff mode. Multiple addresses may be specified in a comma separated list.

Example:

```
<domain *>
    backoff-notify two@host.domain,one@host.domain
</domain>
```

`backoff-to-normal-after`

> **Scope:** domain
>
> **Type:** time {Nd|h|m|s|}
>
> **Attributes:** optional
>
> **Default:** never

Specifies a time interval after which a queue automatically goes back to normal mode. Example:

```
<domain *>
    backoff-to-normal-after 6h
</domain>
```

`backoff-to-normal-after-delivery`

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

If enabled, this puts a queue back into normal mode after a successful delivery.

94

backoff-upon-all-sources-disabled

      Scope: domain

      Type: boolean

      Attributes: optional

      Default: false

If enabled, switches a queue into backoff mode upon all source IPs become disabled, whether via command line command or when using the <smtp-pattern-list> function "disable-source-ip".

Note: Port25 recommends using the "backoff-to-normal-after" directive in conjunction with this directive, since PowerMTA does not automatically put a queue back into normal mode when an IP address reverts back to being enabled.

reenable-source-ip-after

      Scope: domain

      Type: boolean

      Attributes: optional

      Default: never

Sets the interval of time for when an IP disabled with the <smtp-pattern-list> function "disable-source-ip" is to be re-enabled, when no "reenable-after=" option is defined in the pattern action. This directive is ignored however when the "disable-source-ip" actions are defined with the accompanying "reenable-after=" option.

```
cold-virtual-mta
```

Scope: virtual-mta

Type: VirtualMTA-name

Attributes: optional

Default:

This directive takes the name of a manually configured VirtualMTA.   The directive can be repeated to configure multiple cold VirtualMTAs, in which case these are used in round-robin fashion and the per-domain limit applies individually for each cold VirtualMTA.

Example:

```
<virtual-mta vmta2>
  smtp-source-host 1.2.3.4 mail2.yourdomain.com
</virtual-mta>

<virtual-mta vmta3>
  smtp-source-host 5.6.7.8 mail3.yourdomain.com
</virtual-mta>

<virtual-mta vmta1>
  smtp-source-host 2.3.4.5 mail3.yourdomain.com
  cold-virtual-mta vmta2
  cold-virtual-mta vmta3
  <domain *>
    max-cold-virtual-mta-msg 1000/d
  </domain>
</virtual-mta>
```

```
max-cold-virtual-mta-msg
```

Scope: domain

Type: N/d

Attributes: optional

Default: 0/d

A per-domain "max-cold-virtual-mta-msg N/day", default of 0, specifies how many messages go to the "foo-cold" vmta when VirtualMTA "foo" is selected. Re-selection occurs at message reception and over SMTP only. The valued specified is the first X amount of messages for that domain. The defined amount is set for delivery upon injection.

```
<virtual-mta vmta1>
  smtp-source-host 1.2.3.4 vmta1.example.com
  cold-virtual-mta vmta2
    <domain *>
      max-cold-virtual-mta-msg 1000/day
    </domain>
</virtual-mta>

<virtual-mta vmta2>
    smtp-source-host 5.6.7.8 vmta2.example.com
    <domain *>
      max-msg-rate  100/h
    </domain>
</virtual-mta>
```

Please note that this setting is per domain, so 1000 emails will go to yahoo, 1000 will go to hotmail, etc.

Auto increments are supported in the following manner:

```
<domain yahoo.com>
   max-cold-virtual-mta-msg 1/day,2/day,3/day,4/day
</domain>
```

In the example above, all cold vmtas will have 1 as their first day's limit for yahoo.com, 2 as the limit for second day to yahoo.com and so on. There is no restriction on the number of per-day limits that can be configured. The last limit value will continue to be used as the per-day limit after all the preceding limits have been used.

```
backoff-reroute-to-virtual-mta
```

Scope: domain

Type: VirtualMTA name

Attributes: optional

Default: none

Specifies that PowerMTA should reroute messages to the given VirtualMTA in case the queue enters backoff mode. For example, if messages are queued for `example.port25.corm/vmta1` and that queue enters backoff mode while `backoff-reroute-to-virtual-mta` is set to `vmta2`, all of its messages would be rerouted to the queue `example.port25.com/vmta2`. New messages entering the queue are also rerouted, as long as the queue remains in backoff mode.

Example:

```
<virtual-mta vmta1>
    ...
</virtual-mta>

<virtual-mta vmta2>
    ...
</virtual-mta>

<domain *>
    backoff-reroute-to-virtual-mta vmta2
</domain>
```

It may also make sense to use this directive on a per VirtualMTA basis:

```
<virtual-mta vmta1>
    ...
    <domain *>
        backoff-reroute-to-virtual-mta vmta2
    </domain>
</virtual-mta>

<virtual-mta vmta2>
    ...
</virtual-mta>
```

WARNING: PowerMTA performs full loop detection on this directive. If a loop is detected, messages are left in the last queue in the loop, and an error is written to the log file.

A reroute from one VirtualMTA to itself is silently ignored, and a reroute to a VirtualMTA pool is not supported.

Type

> **Scope:** domain
>
> **Type:** `smtp`, `pipe`, `file`, or `discard`
>
> **Attributes:** optional
>
> **Default:** see text

> Change from one type to another with messages in the given queue requires a restart of PowerMTA.

> Determines whether e-mail for this domain is to be delivered over SMTP, through the pipe delivery API, to a file, or discarded.

> The default for the `*` domain is `smtp`, which unless overriden is inherited by all other domains.

> File delivery writes messages to files, just like the Pipe API sample programs `appendtofile` and `newfile` (see [Section 7.5.2](#)). The difference is that while the built-in file delivery is faster, pipe delivery is more flexible (since it allows you roll your own delivery programs). See [Section 3.2.11](#) for file delivery-specific directives.

> Discard delivery discards the e-mail, after writing the corresponding accounting record and including the e-mail in the traffic counters. Discard delivery is useful if you are only interested in the accounting records.

include-headers-from

> **Scope:** virtual mta,global
>
> **Type:** file name
>
> **Attributes:** optional
>
> **Default:** none

> Specifies the name of a file containing RFC2822-formatted headers to insert in all messages delivered from a VirtualMTA. The headers are inserted immediately after the `Received` header added by PowerMTA. When used globally all VirtualMTAs will inherit the settings. If a VirtualMTA already has the directive defined, both settings will be used. This directive is reloadable.

`domain-macro`

    Scope: global

    Type: list

    Attributes: optional

    Default: none

Specifies the variable and bindings for domain macro expansion. Only letters & numbers are supported when naming a domain-macro. See the domain scope in Section 3.2 for more information on usage.

`accept-invalid-recipients`

    Scope: source

    Type: boolean

    Attributes: optional

    Default: no

Allows feeding invalid recipients into PowerMTA (which are then immediately bounced). Invalid in this sense means syntactically illegal email addresses (i.e., me@there or you@yahoo..com), but not e.g., bad mailboxes (syntactically correct but non-existent email addresses).

Pickup directory is not supported. For this functionality in a pickup file use XACK OFF/ON. See section 7.4.1 for more information.

virtual-mta

>**Scope:** virtual mta pool
>
>**Type:** VirtualMTA name
>
>**Attributes:** optional
>
>**Default:** none

Specifies the name of a VirtualMTA to include in the pool. To specify multiple VirtualMTAs, repeat this directive with the name of each VirtualMTA to add.

Example:

```
<virtual-mta mta1>
   ...
</virtual-mta>

<virtual-mta mta2>
   ...
</virtual-mta>

<virtual-mta-pool pool>
    virtual-mta mta1
    virtual-mta mta2
</virtual-mta-pool>
```

3.3.9 SMTP Delivery Directives

The directives in this Section are ignored if specified in non-SMTP domains.

allow-cancel-during-transfer

>**Scope:** domain
>
>**Type:** boolean
>
>**Attributes:** optional
>
>**Default:** true

Specifies whether PowerMTA should allow connections to this domain to be cancelled (e.g., in order to shut down PowerMTA) while a message's contents (headers and body) are being transferred. Some mailers have been known to still deliver messages if the connection breaks before they are fully received, lastly causing partial and duplicate messages to be delivered as PowerMTA retries delivering. Setting this directive to `false` prevents PowerMTA from cancelling connections while messages are being transferred on it, but also possibly causes PowerMTA to take longer to shut down.

auth-username

> **Scope:** domain
> **Type:** string
> **Attributes:** optional
> **Default:** none

Specifies the user name to use when authenticating to the remote mailer.

Authentication is attempted if the `AUTH` extension as well as a SASL authentication mechanism supported by PowerMTA are available at the remote mailer, and either `auth-username` or `auth-password` have non-empty values.

PowerMTA currently only supports the `CRAM-MD5, LOGIN, & PLAIN` authentication mechanisms.

smtp-data-termination-timeout

> **Scope:** domain
> **Type:** time
> **Attributes:** optional
> **Default:** 10m

Allows for setting the amount of time that PowerMTA will wait for the final "250 ok" response after sending the message body. Setting this value too low is likely to cause duplicates.

assume-delivery-upon-data-termination-timeout

> **Scope:** domain
> **Type:** Boolean
> **Attributes:** optional
> **Default:** no

If enabled and the connection times out while waiting for the reply to the "." after a DATA or for the reply to the last BDAT segment, PowerMTA assumes that the delivery was successful.

auth-password

> **Scope:** domain
> **Type:** string
> **Attributes:** optional
> **Default:** none

Specifies the password to use when authenticating to the remote mailer. Please see the `auth-username` directive above for more information.

102

use-unencrypted-plain-auth

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should use PLAIN and/or LOGIN for authentication in case the connection is not encrypted (i.e., if STARTTLS is not being used).

smtp-553-means-invalid-mailbox

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** yes

Instructs PowerMTA whether to default to a 5.1.1 upon a 553 reply to a RCPT command and no enhanced status code is provided in the reply.

smtp-421-means-mx-unavailable

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** no

If set to "yes", PowerMTA will immediately break the connection upon receiving a 421 error and subsequently attempt a new connection to the next MX record listed in DNS. The default setting of "no" will cause PowerMTA to reset the connection and then reuse the same connection for additional recipients.

replace-smtp-421-service-response

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

When enabled, this replaces the SMTP response after the 421 code and after any enhanced status code with "service unavailable", thus preventing any email addresses, etc. that may be included in that response from being used in bounces. The original message is still passed in for pattern matching, so that these continue to work normally. Defaults to false.

103

```
bounce-upon-no-mx
```

Scope: domain

Type: boolean or time interval

Attributes: Optional

Default: False

If set to "true" and no MX records are found for the domain, all emails in the queue bounce immediately. Note that this applies only to cases where a MX lookup is performed, which does not include cases where a host list is specified in the "route" directive (unless it contains only domain name, in which case a MX lookup is performed).

When a time interval is defined instead, for example, 30m, it defines the time interval to bounce messages from the queue if the last DNS query resulted in no MX records being returned. Defining a time interval allows for more tolerance in dealing with domains that do not have MX records, since some sites still accept email on their A records. Note that this applies only to cases where a MX lookup is performed. It does not include cases where an IP list is specified in the "route" directive (unless it contains only a domain name, in which case a MX lookup is performed on the domain).

Example:

```
<domain example.com>
    bounce-upon-no-mx true
</domain>

<domain *>
    bounce-upon-no-mx 30m
</domain>
```

```
bounce-upon-5xx-greeting
```

Scope: domain

Type: boolean

Attributes: optional

Default: true

Specifies whether PowerMTA should immediately bounce all messages for the domain when receiving a 5xx SMTP greeting. E-mail protocols mandate this default behavior, however there exist many misconfigured mailers on the network, causing e-mail to be lost unnecessarily. If this directive is set to `false`, PowerMTA will regard 5xx greetings as a temporary error and try delivering the messages to any secondary MX mailers, as well continue retrying delivery until the `bounce-after` interval expires.

bounce-upon-pix-transfer-failure

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

This directive is identical to the `bounce-upon-transfer-failure` directive below, except that the message is only bounced if a Cisco PIX Firewall is detected at the receiving site. Detection is performed based on the SMTP greeting line: if it contains at least ten asterisks, a PIX firewall is detected.

bounce-upon-transfer-failure

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should immediately bounce a message whose delivery failed while its contents (message headers and body) were being transferred. Normally, PowerMTA would not immediately bounce the message, but continue trying until it becomes older than the `bounce-after` interval. Due to deficiencies in the SMTP protocol (See RFC 1047) and in the receiving mail servers, if the TCP connection fails while the contents are being transmitted, there is some risk that various copies of the same message will be delivered, some of which possibly with only partial contents. By enabling this directive, you instruct PowerMTA to bounce the message immediately after the first failed transfer attempt. This effectively prevents sending out duplicates, but may also cause the recipient to never receive the message, or receive only one (possibly partial) copy of it.

reject-mfrom-check-temperror

> **Scope:** source
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** true

Specifies whether to reject the MAIL command (with a 4xx error) in the SMTP protocol if the "mfrom" check results in a temporary error, such as when unable to look up certain DNS records.

reject-mfrom-check-permerror

     Scope: source

     Type: boolean

     Attributes: optional

     Default: true

Specifies whether to reject the MAIL command (with a 5xx error) in the SMTP protocol if the "mfrom" check results in a permanent error, such as when there is a syntax error in a SPF record.

reject-mfrom-check-fail

     Scope: source

     Type: boolean

     Attributes: optional

     Default: true

Specifies whether to reject the MAIL command (with a 5xx error) in the SMTP protocol if the "mfrom" check results in a failure, such as when email is being sent from an unauthorized IP address.

connect-timeout

     Scope: domain

     Type: time interval

     Attributes: optional

     Default: 2m (two minutes)

Specifies the maximum amount of time that PowerMTA waits for an outgoing SMTP connection to be established.

smtp-greeting-timeout

     Scope: domain

     Type: time interval

     Attributes: optional

     Default: 5m (five minutes)

Specifies the amount of time to wait for the initial SMTP 220 greeting message to be received after the connection is accepted. Previously PowerMTA waited up to 10 minutes for the greeting message.

data-send-timeout

> **Scope:** domain
>
> **Type:** time interval
>
> **Attributes:** optional
>
> **Default:** 3m (three minutes)

Specifies the maximum amount of time that PowerMTA waits for a chunk of data (message contents and body) to be sent over an outgoing SMTP connection. For full compliance with RFC 2821, this timeout needs to be at least three minutes.

ignore-8bitmime

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should ignore the 8BITMIME extension even when supported by the receiving mailer. Setting this directive to true allows you to work around the problem of messages bouncing because a site's external MTA accepting 8BITMIME messages as such but unable to convert them to 7BIT when passing to an internal MTA which does not support 8BITMIME.

The directive doesn't change the encoding of the message, but rather, by default if a message has 8bit encoding and the feeding application passed that information along in the MAIL FROM command, PowerMTA passes that info along to the remote mail server via the MAIL FROM command. Setting the value to true tells PowerMTA to not pass along the information. This has nothing to do with anything declared in the body of the message. See RFC 1652 for more information http://www.ietf.org/rfc/rfc1652.txt.

ignore-chunking

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should ignore the CHUNKING extension even when supported by the receiving mailer. Setting this directive to true may be useful for debugging message transfer problems.

`ignore-pipelining`

> **Scope:** domain
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether PowerMTA should ignore the `PIPELINING` extension even when supported by the receiving mailer. Setting this directive to `true` may be useful for debugging message transfer problems.

`max-msg-per-connection`

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** 0

Specifies the maximum number of messages delivered within a single connection. Normally it is most efficient to deliver as many messages as possible per connection, but in special circumstances it may be desirable to have PowerMTA close the connection and reconnect before delivering more messages. A value of zero means no limit is imposed.

`max-rcpt-per-message`

> **Scope:** domain, source
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** 1000

Alias to `max-rcpt-per-transaction.`

`max-rcpt-per-transaction`

     Scope: domain

     Type: number

     Attributes: optional

     Default: 1000

Specifies the maximum number of recipients passed with each message. For example, if a message is submitted including 5,000 recipients for a single domain, and `max-rcpt-per-message` for that domain is set to 3,000, PowerMTA will transfer the message twice, once for 3,000 recipients and again for the remaining 2,000 recipients. Smaller settings increase parallelism by allowing the same message to be transferred over several connections to the same domain, but also reduce efficiency, since the message contents are transferred several times. `max-rcpt-per-transaction` is an alias to `max-rcpt-per-message`.

`max-smtp-out`

     Scope: domain, virtual-mta

     Type: number

     Attributes: optional

     Default: 20 for domain, unlimited for virtual-mta

Specifies the *maximum* number of simultaneous connections to be opened `for this domain or VirtualMTA`. The actual number of connections opened is determined by PowerMTA depending on a variety of parameters and in practice rarely exceeds one or two for an individual domain.

Certain circumstances may warrant increasing this number for specific domains, however many sites will have an unpublished maximum number of connections allowed at a time, and which can and does change from time to time. So, while increasing this number for specific domains could improve throughput performance, that is not always the case and ultimately you have to play by the rules imposed by the receiving mail servers.

`backoff-max-smtp-out`

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** 20

Specifies the *maximum* number of simultaneous connections to be opened `for this domain` when in backoff mode. The actual number of connections opened is determined by PowerMTA depending on a variety of parameters and in practice rarely exceeds one or two for an individual domain.

`max-smtp-out-per-source-ip`

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** unlimited

Deprecated, use source-ip-max-smtp-out instead

source-ip-max-smtp-out

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** unlimited

Specifies the *maximum* number of simultaneous connections to be opened for this domain for a given source IP. The actual number of connections opened is determined by PowerMTA depending on a variety of parameters and in practice rarely exceeds one or two for an individual domain.

Certain circumstances may warrant increasing this number for specific domains, however many sites will have an unpublished maximum number of connections allowed at a time, and which can and does change from time to time. So, while increasing this number for specific domains could improve throughput performance, that is not always the case and ultimately you have to play by the rules imposed by the receiving mail servers.

```
<domain comcast.net>
  source-ip-max-smtp-out 1.2.3.4  10
</domain>
```

`max-connect-rate`

> **Scope:** domain
>
> **Type:** {unlimited|0|N/{h|hr|m|min|s|sec}}
>
> **Attributes:** optional
>
> **Default:** unlimited

Specifies the *maximum* number of connections to be opened for this domain during the specified time period.

`source-ip-max-connect-rate`

> **Scope:** domain
>
> **Type:** {unlimited|0|N/{h|hr|m|min|s|sec}}
>
> **Attributes:** optional
>
> **Default:** unlimited

Specifies the *maximum* number of connections to be opened for this domain during the specified time period per IP per domain/vmta.

```
<domain comcast.net>
  source-ip-max-connect-rate 1.2.3.4 60/m
</domain>
```

`source-ip-max-msg-rate`

> **Scope:** domain
>
> **Type:** {unlimited|0|N/{h|hr|m|min|s|sec}}
>
> **Attributes:** optional
>
> **Default:** unlimited

Specifies the *maximum* number of messages to be attempted for delivery for this domain during the specified time period per IP per domain/vmta.

```
<domain comcast.net>
  source-ip-max-msg-rate 1.2.3.4 600/m
</domain>
```

`backoff-max-connect-rate`

> **Scope:** domain
>
> **Type:** {unlimited|0|N/{h|hr|m|min|s|sec}}
>
> **Attributes:** optional
>
> **Default:** unlimited

Specifies the *maximum* number of connections to be opened for this domain during the specified time period when in backoff mode.

retry-upon-new-mail

>**Scope:**   domain
>
>**Type:**   boolean
>
>**Attributes:** optional
>
>**Default:**   false

Specifies if PowerMTA should immediately schedule delivery of a message when it is received if the queue to which it is injected is already in retry mode. If set to true and a queue is in retry mode, new mail injected into that queue will cause the queue to be scheduled for delivery.

pix-bug-workaround

>**Scope:**   domain
>
>**Type:**   boolean
>
>**Attributes:** optional
>
>**Default:**   true

Some versions of the Cisco PIX firewall have a bug (bug ID CSCds90792) in its "fixup smtp" command in that it rejects e-mail (causing duplicates to be delivered) if the final .`<CR><LF>` sequence is split across various TCP frames. This directive instructs PowerMTA, in case it detects a PIX firewall, to send the final .`<CR><LF>` in a frame of its own, making it almost certain to arrive at the receiving site in a single frame. Detection is performed based on the SMTP greeting line: if it contains at least ten asterisks, a PIX firewall is detected.

```
route
```

Scope: domain
Type: string
Attributes: optional
Default: none

Allows you to override the normal DNS-based routing, instructing PowerMTA to deliver to the specified host name or IP address. Multiple hosts can be specified in a comma and semicolon-separated list. MX record lookup is only supported if only one host name is specified, otherwise the A/AAAA record is retrived. The format of the route string is one of the following examples.

The following specifies a host name, whose IP address will be looked up in the DNS.

```
<domain SomeDomain.com>
    route OtherDomain.com
</domain>
```

The following specifies an IP address to which PowerMTA is to connect.

```
<domain SomeDomain.com>
    route [a.b.c.d]
</domain>
```

The following specifies an IP address and a TCP port number to which PowerMTA is to connect.

```
<domain SomeDomain.com>
    route [a.b.c.d]:p
</domain>
```

Lastly, the following is an example where commas separate hosts within the same priority level, and semicolons separate priority levels. In this example, [a.b.c.d] and [e.f.g.h] are the primary, and equal, hosts to contact in round-robin fashion. [i.j.k.l] is a backup host for [a.b.c.d] and [e.f.g.h], whereas [m.n.o.p]:p is a backup host for [i.j.k.l].

```
<domain SomeDomain.com>
    route [a.b.c.d]:25,[e.f.g.h]:25;[i.j.k.l]:2525;[m.n.o.p]:25
</domain>

To put the above in a chart format:
 Priority     Host
------------------
   0       [a.b.c.d]
   0       [e.f.g.h]
   1       [i.j.k.l]
   2       [m.n.o.p]
```

smtp-hosts

> **Scope:** domain
> **Type:** string
> **Attributes:** optional
> **Default:** none

Same as the route directive but when only one host name is specified it allows looking up the MX record instead of just an A/AAAA record by adding the lookup-mx: prefix. If more than one host name is specified all the host names will only have their A/AAAA record retrieved.

```
<domain *>
  smtp-hosts lookup-mx:example.com
</domain>
```

mx-connection-limit

> **Scope:** global
> **Type:** string
> **Attributes:** optional
> **Default:** none

Per connection limits can be configured with the directive:

```
mx-connection-limit mx.domain.com limit
```

In the above example, "mx.domain.com" is the MX record that you want to limit connections to, and "limit" is either a positive number or the word "unlimited". One can define a specific MX record to match on, or use wildcards to define multiple MX records, for example, [*.]domain.com, using the same conventions available within <domain> tags. The connection limit applies on a per source IP address basis (per IP address that PowerMTA will make connections from). If the wildcard is used, and multiple records are matched, the connection limit will apply per record matched, and not globally across all matches.

The global directive can be entered several times in the configuration file, once per line, forming a list of rules specifying the limits for the various MX records. It is important to note that mx.domain.com parameter matches the MX record for a domain, and not the recipient domain itself.

This directive is most useful when an ISP or hosting provider hosts hundreds of domains, but has one or few MX records for all of those domains. This directive allows you to limit the number of connections per MX, regardless of the number of recipient domains you are mailing to. Note again, that the connection limit applies on a per source IP address basis and not per VirtualMTA.

```
mx-connection-limit [*.]secureserver.net 20
```

ip-connection-limit

> **Scope:** global
>
> **Type:** string
>
> **Attributes:** optional
>
> **Default:** none

Per connection limits can be configured with the directive:

```
ip-connection-limit IP limit
```

The directive is only permitted in global scope. The syntax for the directive is similar to mx-connection-limit directive, except that it is uses IP addresses/CIDR notation.For example:

```
ip-connection-limit 10.0.0.0/8 5
```

Any remote IP addresses(the A records for the remote MX) which map to 10.0.0.0/8 subnet will be subject to the limit of 5 simultaneous connections at any point in time.

default-smtp-port

> **Scope:** domain
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** 25

Allows specifying the TCP port to use for delivering over SMTP (unless overridden by a "route" directive). This allows people to use the destination port to pass on a number (namely, the port number) to their NAT boxes, thus simplifying MTA farm setup (because one internal IP address is no longer needed per vmta per MTA box).

smtp-pattern-list

> **Scope:** domain
>
> **Type:** SMTP pattern list name
>
> **Attributes:** optional
>
> **Default:** none

Specifies that the given SMTP pattern list is to be used in connections to the given domain. The SMTP pattern list referenced must precede the domain definition in the configuration file. Section 3.2.17 describes SMTP pattern lists in greater detail.

115

`smtp-source-host`

Scope: global, virtual-mta

Type: IP address or CIDR range and host-name

Attributes: optional

Default:

On a multi-homed host, this directive specifies an IP address (or IP address range) and host-name from which PowerMTA is to initiate outgoing SMTP connections. This can be useful, for example, to help ensure that PowerMTA complies with your firewall policy. Alternatively, you can enter this directive several times and/or specify a CIDR IP address range with the host-name, instructing PowerMTA to use the various source IP addresses in round-robin fashion. All local IP addresses matching the specification are added to the list of source IP addresses, except for 127.0.0.1, which is automatically excluded since it can't be used to connect to remote hosts.

When specified within a VirtualMTA, this directive overrides the global source IPs for connections established to send messages through the VirtualMTA.

These IP/host-name pair should match in DNS as the IP/PTR (reverse DNS) pair. When multiple pairs are used, any <domain> settings are applied across all pairs, and not set individually. For example, in the following:

```
<virtual-mta customer1>
  smtp-source-host 192.168.0.10 mail10.yourserver.com
  smtp-source-host 192.168.0.20 mail20.yourserver.com
  smtp-source-host 192.168.0.30 mail30.yourserver.com
  <domain yahoo.com>
    max-smtp-out 12
  </domain>
</virtual-mta>
```

The setting "max-smtp-out 12" will result in no more than 12 connections across all three pairs together at one time, not 12 each for a total of 36. Control of individual pairs is not currently available, and if needed, separate VirtualMTAs can be configured for each pair.

This may be used in the place of a <virtual-mta-pool> to help with issues around backoff mode when using backoff-reroute-to-virtual-mta. Using the above instead will keep all the messages in the same VirtualMTA (for better control), and allow all defined IPs to be used.

assume-delivery-upon-data-termination-timeout

> **Scope:**   domain
> **Type:**   boolean
> **Attributes:** Optional
> **Default:**   false

If enabled and the connection times out while waiting for the reply to the "." after a DATA or for the reply to the last BDAT segment, PowerMTA assumes that the delivery was successful. .May result in duplicate messages if used when not needed.

3.3.9.1 SSL/TLS Directives

use-starttls

> **Scope:**   domain
> **Type:**   boolean
> **Attributes:** Optional
> **Default:**   true

Specifies whether PowerMTA should use the STARTTLS extension (if supported by the receiving mailer). Setting this directive to true instructs PowerMTA to use that extension to encrypt the connection to the remote mailer, at a cost in performance.

require-starttls

> **Scope:**   domain
> **Type:**   boolean
> **Attributes:** Optional
> **Default:**   false

Only relevant if "use-starttls" is enabled. The STARTTLS extension may not be available at the remote host, or its use may fail for various reasons. If "require-starttls" is set to "true", PowerMTA will abort the connection, thus avoiding to send the message through a less secure connection. Otherwise, it will send the message anyway.

smtp-tls-allow-tlsv1

Scope: domain

Type: boolean

Attributes: Optional

Default: true

Allows PowerMTA to use TLSv1 when establishing a secure connection. To workaround an issue with outdated IronPort servers, the required change is:

```
<domain example.com>
    smtp-tls-allow-tlsv1 false
</domain>
```

smtp-tls-allow-tlsv2

Scope: domain

Type: boolean

Attributes: Optional

Default: true

Allows PowerMTA to use TLSv1 when establishing a secure connection. To workaround an issue with outdated IronPort servers, the required change is:

```
<domain example.com>
    smtp-tls-allow-tlsv2 false
</domain>
```

smtp-tls-allow-sslv3

Scope: domain

Type: boolean

Attributes: Optional

Default: true

Allows PowerMTA to use SSLv3 when establishing a secure connection.

118

smtp-tls-ca-file directive

> **Scope:** domain
>
> **Type:** string
>
> **Attributes:** Optional
>
> **Default:** none

Specifies the location of a local text file containing one or more root certificates that will be used to verify remote server certificate chains. Used by senders to authenticate remote gateways when using TLS based delivery. If using opportunistic TLS and the certificate verification fails, PowerMTA will fall back to using plain text for the delivery. In contrast, if TLS is required and the certificate verification fails, PowerMTA will return a temporary 4xx error indicating the failure. Example**:**

```
<domain bounces.xyz.com>
    smtp-tls-ca-file /etc/pmta/certchain.pem
</domain>
```

3.3.10 Discard Delivery Directives

Currently there are no Discard Delivery-specific directives.

3.3.11 File Delivery Directives

The directives in this Section are ignored if specified in non-File Delivery domains.

file-format

> **Scope:**   domain
>
> **Type:**   one of: `newfile-plain`, `newfile-pickup` or `append-mbox`
>
> **Attributes:** optional
>
> **Default:**   `newfile-plain`

Specifies the format of the file(s) to which PowerMTA delivers the messages:

newfile-plain
> A new file is created for each message, in the directory indicated by `file-destination`. The entire message is written to the file, including all headers, but no envelope information.

newfile-pickup
> A new file is created for each message, in the directory indicated by `file-destination`. In addition to the entire message body and headers, the special headers `x-sender` and `x-receiver` are included, containing the originator and recipient addresses from the message envelope.

append-mbox
> Messages are appended to the file indicated by `file-destination`, in the unix "mbox" format.

Like with the sample Pipe Delivery applications, special care must be taken when accessing files to which PowerMTA performs file delivery.

PowerMTA creates new files with the initial "extension" of `.tmp` and renames them to `.msg` once done. If post-processing these files as they are created, you should code to ignore any `.tmp` files, to avoid processing a file which hasn't yet been completely written.

When appending to a file (for `append-mbox`), interlocking is performed using `flock` on unix-like platforms and with share options passed to `CreateFile` on Windows. In order to safely access the file while PowerMTA is running, Port25 recommends using the `pmtagetfile` application. Please see Section 7.5.3 for more information on `pmtagetfile`.

```
file-destination
```

Scope: domain

Type: absolute directory or file name

Attributes: optional

Default: none

Specifies the directory or file name for file delivery. The directory or file name is subjected to the same macro expansion as in the pipe delivery's command line:

`$bodytype`
> The message body type (`7BIT`, `8BITMIME` or `BINARYMIME`);

`$ctime`
> The current date in Unix `ctime` format;

`$envid`
> The DSN envelope ID (from `MAIL ... ENVID=...`);

`$from`
> The envelope sender (`MAIL FROM`) address;

`$isodate`
> The current date in the format YYYY-MM-DD;

`$timet`
> The current time in numeric (`time_t`) format;

`$rfctime`
> The current date in RFC 822 format;

`$to`
> The envelope recipient (`RCPT TO`) address;

`$user`
> The recipient user. That is the local part (mailbox) in the recipient address, or, if the local part contains a '+' character, the portion of the mailbox to the left of the '+'. For example, in `joe+xyzzy@company.com`, the user would be `joe`;

`$domain`
> The recipient's domain;

$from, $to, $user and $domain macros always expand in lower case, avoiding issues when those are used to create directories and files on case-sensitive file systems **Examples:**

```
<domain bounces.xyz.com>
    type file
    file-format newfile-pickup
    file-destination c:/campaigns/bounces
</domain>

<domain [*.]bounces.xyzzy.com>
    type file
    file-format append-mbox
    file-destination /my/path/$domain
</domain>
```

121

3.3.12 Pipe Delivery Directives

The directives in this Section are ignored if specified in non-pipe domains.

command

>   **Scope:**   domain
>
>   **Type:**   string
>
>   **Attributes:** required
>
>   **Default:**   none

Specifies the command to use to start the program which will read from the pipe. This directive is required for pipe domains.

PowerMTA substitutes the following macros in the command line:

$bodytype
>   The message body type (`7BIT`, `8BITMIME` or `BINARYMIME`);

$ctime
>   The current date in Unix `ctime` format;

$envid
>   The DSN envelope ID (from `MAIL ... ENVID=...`);

$from
>   The envelope sender (`MAIL FROM`) address;

$timet
>   The current time in numeric (`time_t`) format;

$rfctime
>   The current date in RFC 822 format;

$to
>   The envelope recipient (`RCPT TO`) address;

$user
>   The recipient user. That is the local part (mailbox) in the recipient address, or, if the local part contains a '+' character, the portion of the mailbox to the left of the '+'. For example, in `joe+xyzzy@company.com`, the user would be `joe`;

$domain
>   The recipient's domain;

To prevent any one of these macros from being interpreted as multiple arguments if they contain spaces, it is a good idea to enclose them in double quotes (like in the example below).

Example:

```
<domain bounces.xyz.com>
    type pipe
    command "/my/bounce/processor --envid \"$envid\" \"$user\""
</domain>
```

122

3.3.13 Submission API Directives

always-allow-api-submission

    Scope: source

    Type: boolean

    Attributes: optional

    Default: no

This directive allows easily enabling all SMTP permissions and features required by our submission APIs. If set to "yes", this is currently equivalent to setting "allow-mailmerge yes". If set to "no", the other settings determine whether submission is allowed.

pickup

    Scope: global

    Type: see text

    Attributes: optional

    Default: none

This directive specifies the path to a "pickup directory" from which PowerMTA processes e-mail submissions, as well as the path to a corresponding "bad mail" directory to which improperly formatted messages are moved. See Section 7.4 for more information on the pickup directory. Only one pickup directory is supported. A restart of the service is required when adding or changing the directive.

See Section 3.2.13.1 for directives that apply to only the messages in the pickup directory.

Example:

```
Pickup  c:\Inetpub\Mailroot\Pickup c:\Inetpub\Mailroot\BadMail
```

pickup-retry-interval

    Scope: global

    Type: time interval

    Attributes: optional

    Default: 5 minutes

Deprecated. PowerMTA will now always scan the pickup directory at the rate of once a second.

3.3.14 Source {Pickup}

For directives that apply only to messages in the pickup directory, use <source {pickup}>
with the following directives. PowerMTA will still accept <source pickup>, but a
warning saying it's deprecated will be returned. Use <source {pickup}> instead.

dsn-return-default

>   **Scope:**   source {pickup}
>
>   **Type:**   `full,` `headers` or `system`
>
>   **Attributes:** optional
>
>   **Default:**   `system`

>   Specifies the default for the DSN `RET` parameter, i.e., whether the full message
>   body or only its headers should be returned in a DSN delivery report. `system`
>   specifies that the system default should be used. This directive is overridden by
>   the use of "dsn-format plain-text".

dsn-format

>   **Scope:**   source {pickup}
>
>   **Type:**   {standard|plain-text}
>
>   **Attributes:** optional
>
>   **Default:**   `standard`

>   The default is "standard", which has PowerMTA send a DSN in the standard
>   format. If set to "plain-text", the "message/delivery-status" portion of the DSN
>   report is delivered instead of the full report. The portion is delivered within a
>   mime-type of "text/plain", thus allowing this data to pass through (overzealous)
>   email firewalls that strip all "attachments"

default-virtual-mta

>   **Scope:**   source {pickup}
>
>   **Type:**   VirtualMTA name
>
>   **Attributes:** optional
>
>   **Default:**   none

>   Specifies the default VirtualMTA to select for all messages received from the
>   source. If `process-x-virtual-mta` is also enabled for the source, the default
>   VirtualMTA can be overriden by means of a `x-virtual-mta` header.

process-x-virtual-mta

    Scope: source {pickup}

    Type: boolean

    Attributes: optional

    Default: false

Specifies whether PowerMTA should process `x-virtual-mta` headers. If set to `true` and one such header is included, PowerMTA will select the VirtualMTA specified in the header's body for the message, as well as remove the header from the message. If set to `false`, PowerMTA will ignore this header, leaving it in the message if present.

process-x-envid

    Scope: source {pickup}

    Type: boolean

    Attributes: optional

    Default: true

Specifies whether PowerMTA should process `x-envid` headers. If set to `true` and one such header is included, PowerMTA will set the message's (DSN) envelope ID to the header's body, as well as remove the header from the message. If set to `false`, PowerMTA will ignore this header, leaving it in the message if present.

process-x-job

    Scope: source {pickup}

    Type: boolean

    Attributes: optional

    Default: true

Specifies whether PowerMTA should process `x-job` headers. If set to `true` and one such header is included, PowerMTA will set the job ID for the message to the job ID given in the `x-job` header. The job ID must not include any non-printable or white space characters. A message is supposed to include at most one `x-job` header. If using Mailmerge, you must use the *jobid variable, and not the x-job header.

add-message-id-header

>**Scope:**   source {pickup}
>
>**Type:**   boolean
>
>**Attributes:** optional
>
>**Default:**   false

>Specifies whether PowerMTA should add a `Message-Id` header if missing. If a `Message-Id` header is present, it is not overridden.

add-received-header

>**Scope:**   source {pickup}
>
>**Type:**   boolean
>
>**Attributes:** optional
>
>**Default:**   true

>Specifies whether PowerMTA should add a `Received` header upon reception of an email.

pattern-list

>**Scope:**   source {pickup}
>
>**Type:**   pattern list name
>
>**Attributes:** optional
>
>**Default:**   none

>Specifies that messages received from the source are to be matched against the given pattern list. The pattern list referenced must precede the source definition in the configuration file. Section 3.3.16 describes pattern lists in greater detail.

`hide-message-source`

>**Scope:**   source {pickup}
>
>**Type:**   boolean
>
>**Attributes:** optional
>
>**Default:**   false

Specifies that PowerMTA should attempt to hide the source of the message while delivering e-mail from this source. Currently, this just means that the name and IP address of the MTA from which PowerMTA received this message will not be included in the `Received:` header added. Hiding the message's source may be desirable, for example, for security purposes, to avoid revealing details from the internal network from which the message was submitted.

`pickup-remove-dot`

>**Scope:**   global
>
>**Type:**   Boolean
>
>**Attributes:** optional
>
>**Default:**   false

When using SMTP feeding and a line starts with a period, that period needs to be escaped by adding a second period. When using a pickup file, the period does not need to be escaped. If the feeding application is adding two periods to a pickup file this directive may be set to true to allow the second period to be removed for proper formatting of emails.

3.3.15 Dummy SMTP ("blackholing") Directives

`dummy-smtp-ip`

>**Scope:**   global
>
>**Type:**   IP address
>
>**Attributes:** optional
>
>**Default:**   all local IP addresses

Specifies the IP address on which PowerMTA is to listen for incoming connections that are to be handled as "dummy" SMTP connections.

```
dummy-smtp-port
```
> **Scope:** global
>
> **Type:** number
>
> **Attributes:** optional
>
> **Default:** 0 (disabled)

Specifies the TCP port on which PowerMTA is to listen for incoming connections that are to be handled as "dummy" SMTP connections.

```
dummy-smtp-await-slot
```
> **Scope:** global
>
> **Type:** boolean
>
> **Attributes:** optiona
>
> **Default:** false

Specifies whether PowerMTA should wait for a new connection slot when a new dummy SMTP connection request is received but no more slots are available. If set to false, PowerMTA responds with a `421` greeting, indicating that the service is not available, and closes the connection.

```
dummy-smtp-has-chunking
```
> **Scope:** global
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether dummy SMTP connections are to support the CHUNKING SMTP extension.

```
dummy-smtp-has-pipelining
```
> **Scope:** global
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

Specifies whether dummy SMTP connections are to support the PIPELINING SMTP extension.

```
dummy-smtp-has-verp
```
      Scope: global

      Type: boolean

      Attributes: optional

      Default: false

Specifies whether dummy SMTP connections are to support the VERP SMTP extension.

```
dummy-smtp-latency
```
      Scope: global

      Type: number

      Attributes: optional

      Default: 0

Specifies the network latency, in milliseconds, to be simulated in dummy SMTP connections.

```
dummy-smtp-update-stats
```
      Scope: global

      Type: boolean

      Attributes: optional

      Default: true

Specifies whether dummy SMTP connections should update the traffic statistics displayed by e.g. the web-based Status Monitor.

3.3.16 Pattern List Directives

Pattern lists allow you to select a VirtualMTA based on the message's originator, the message's recipient, a message header, or a combination of the three.

The directives described in this section define a list of regular expression patterns. Each entry in a pattern list has the following form:

```
directive parameters /pattern/  virtual-mta=name,recipient-priority=n
```

where:

- *directive* is one of the directives below, specifying what is matched.

- *parameters* additional information depending on the chosen directive; currently only the *header* directive supports a parameter, which is the message header you want to match.
- *pattern* is a perl-compatible regular expression.
- *name* is the VirtualMTA selected.
- *n* is the priority of the recipient using a scale of 0-100 with 0 being the lowest priority and 100 being the highest with a default of 50 when not specified. This may be used without selecting a VirtualMTA.

If multiple entries match, the first match is used.

An `x-virtual-mta` header may be used in conjunction with a <pattern-list> to set a `recipient-priority`.

For more information on perl-compatible regular expressions, please refer to the [Perl documentation](#) or the [PCRE library package](#).

header

    Scope: pattern list

    Type: see text

    Attributes: optional

    Default: none

Adds a pattern match against the specified header. Matching is performed against the entire header contents; in particular, any *white space* characters such as blanks, tabs and CR/LFs will still be present. Because of this, when anchoring the regular expression with the caret (`^`) and dollar (`$`) characters, it is important to allow for white space following the caret and preceding the dollar signs, respectively. The `\s` metacharacter is particularly useful for this, since it will match all white space characters.

Example:

```
<pattern-list myList>
    header x-eval   /^\s*eval/     virtual-mta=evaluation
    header from   /jsmith/     virtual-mta=evaluation
    header subject   /hello world/     virtual-mta=evaluation
</pattern-list>

<source 127.0.0.1>
    ...
    pattern-list myList     # this selects the pattern list for messages
</source>                   # from 127.0.0.1
```

130

mail-from

    Scope: pattern list

    Type: see text

    Attributes: optional

    Default: none

Adds a pattern match against the message's originator. This is the SMTP MAIL FROM address and is the address to which bounces are sent. This is not the same as the from header seen in mail clients.

Example:

```
<pattern-list myList>
    mail-from /^customerservice/    virtual-mta=service
</pattern-list>

<source 127.0.0.1>
    ...
    pattern-list myList     # this selects the pattern list for messages
</source>                   # from 127.0.0.1
```

rcpt-to

    Scope: pattern list

    Type: see text

    Attributes: optional

    Default: none

Adds a pattern match against the message's recipient.

Example:

```
<pattern-list myList>
    rcpt-to /customer/    virtual-mta=customers
</pattern-list>

<source 127.0.0.1>
    ...
    pattern-list myList     # this selects the pattern list for messages
</source>                   # from 127.0.0.1
```

If multiple recipients are specified in a message and more than one match a `rcpt-to` directive, the first match's VirtualMTA will apply to the entire message. This behavior may change in the future, so `rcpt-to` should only be used if you only intend to submit single-recipient messages.

Another example, showing a combination of matching patterns:

```
<pattern-list myList>
    header from      /jsmith/                virtual-mta=evaluation
    header subject   /hello world/           virtual-mta=evaluation
    header x-eval    /eval/                  virtual-mta=evaluation
    mail-from        /newsletter/            virtual-mta=newsletters
    rcpt-to          /^stud.*@somesite.com$/ virtual-mta=students
    rcpt-to          /password-reset@customer.com/  recipient-priority=90
    rcpt-to          /special@customer.com/  virtual-mta=students,recipient-priority=90
</pattern-list>

<source 127.0.0.1>
    ...
    pattern-list myList     # this selects the pattern list for messages
</source>                   # from 127.0.0.1
```

Here, each email is first checked for the `x-eval` header. If it is present and its content contains *eval*, the *evaluation* VirtualMTA is used. If the `x-eval` header isn't given or does not match the pattern, the `mail-from` is checked to contain `newsletter`. If it does, the `newsletters` VirtualMTA is used. Finally, if no VirtualMTA was selected by the two matches, the `rcpt-to` is checked. If it starts with `stud` and ends in `@somesite.com`, the `students` VirtualMTA is used. If none of the above matched, PowerMTA will use the default VirtualMTA.

This example shows conditional pattern matching:

```
<pattern-list patterns>
   mail-from /newsletter/ <pattern-list>
       rcpt-to /^stud.*@somesite.com$/ virtual-mta=students
       rcpt-to /^staff.*@somesite.com$/ virtual-mta=staff
       * virtual-mta=anything
   </pattern-list>

   rcpt-to /admin/ <pattern-list>
       mail-from /thisdomain.com/ virtual-mta=this
       header x-info /newdata/ virtual-mta=other
       * virtual-mta=anything
   </pattern-list>
</pattern-list>

<source 127.0.0.1>
    ...
    pattern-list patterns   # this selects the pattern list for messages
</source>                   # from 127.0.0.1
```

Here, first the MAIL FROM is checked for each email. If the MAIL FROM matches "newletter", a sub-pattern is then used. In this sub-pattern, the RCPT TO is checked, and if it matches the pattern, the emails will go to the "students" VirtualMTA, or the "staff" VirtualMTA. Anything with "newsletter" in the MAIL FROM that does not match the sub-patterns get sent to the wildcard defined "anything" VirtualMTA.

The second pattern is similar to the above, but first first checks the RCPT TO. If the RCPT TO matches, the email is then put through the sub patterns, first checking the

132

MAIL FROM, and then for the "x-info" header. Once again, emails with a RCPT matching "admin", but not the sub-patterns will be sent to the VirtualMTA "anything".

Emails not matching any patterns will not be sent to a VirtualMTA based on pattern matching.

3.3.16.1 Nested pattern-list support

In some cases it may be better to use a <pattern-list> to select another <pattern-list> instead of selecting a VirtualMTA directly. This can be accomplished in PowerMTA with the new "call-to" option defined within a pattern.

Assuming a message has the following header:

```
x-custom-header: abc123
```

and you wanted to use that header and content to select another pattern to then select the correct VirtualMTA. The following would be defined in PowerMTA's config file:

```
#
<pattern-list specialSelector1>
  mail-from /newsletters/ virtual-mta=vmta1
</pattern-list>

<pattern-list selections>
  header x-custom-header /abc123/ call-to: specialSelector1
</pattern-list>

<source 0/0>
  pattern-list selections
</source>
#
```

In the above example, <pattern-list selections> would match the x-custom-header: header, and then select <pattern-list specialSelector1> to then apply next to the message, given the "call-to:" definition. If the SMTP MAIL FROM address for this message included "newsletters", PowerMTA would then use VirtualMTA vmta1 for the message.

RFC 2047, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", allows for non-ASCII characters to be used in headers. In the event that these characters are needed to be used in a header that is also used for pattern matching, PowerMTA has the ability to pattern match on these characters.

3.3.17 SMTP Pattern List Directives

SMTP pattern lists allow you to have PowerMTA change a delivery queue's mode based on pattern matching on the SMTP replies received while attempting deliveries.

SMTP pattern lists currently only accept a single directive (`reply`). Each entry in a pattern list has the following form:

```
reply /pattern/  action
```

Where:

- *pattern* is a perl-compatible regular expression.
- *action* is one or a comma separated list of the following
    - mode=`normal`
    - mode=`backoff`
    - skip-mx
    - bounce-queue
    - bounce-rcpt
    - disable-source-ip
    - reenable-after
    - defer-queue

If multiple conflicting entries match, the first match is used. The following is not allowed:

- both `mode=backoff` and `mode=normal`
- both `skip-mx` and `bounce-queue`
- `reenable-after` without a preceding `disable-source-ip` (in the same rule)
- more than one `reenable-after`.

For more information on perl-compatible regular expressions, please refer to the [Perl documentation](#) or the [PCRE library package](#).

skip-mx is intended to help PowerMTA from give up delivering to one or more MXes which is just giving a transient recipient-level error. Any patterns matched will result in the next MX being used and mail being held in queue, even in the case of a 5xx error.

bounce-queue will bounce the entire queue immediately if the pattern is matched. The original error, as well as the fact that the queue was bounced due to this setting, will be recorded in the accounting file.

bounce-rcpt will bounce the recipient immediately if the pattern is matched. This only works for 4xx responses to the RCPT command. The original error, as well as the fact that the recipient was bounced due to this setting, will be recorded in the accounting file. This will bounce a recipient even if "smtp-421-means-mx-unavailable true" is set.

disable-source-ip will disable the IP that received the error. The source IP will remain disabled until the time period defined by the <domain> directive `reenable-source-ip-after`, unless using the `reenable-after` option.

reenable-after Allows for defining the time after which an IP that has been disabled with disable-source-ip will be reenabled. The default is never. Takes priority over the domain's `reenable-source-ip-after` setting when used.

defer-queue When a queue goes in backoff mode, it does not automatically get put into retry mode as well. Not all SMTP errors will have PowerMTA putting the queue into retry mode immediately vs., for example, having PowerMTA reconnect on the next MX. Since it may be desirable to have the queue put into retry mode immediately for certain errors, a new defer-queue option was added in SMTP pattern lists. This option gives you full control in forcing the queue into retry mode immediately if desired, on a per SMTP response basis.

For example, to put any queues in backoff mode upon reception of certain SMTP replies, while bouncing some messages immediately or skipping some MX records:

```
<smtp-pattern-list blockList>
    reply /generating high volumes of.* complaints from AOL/    mode=backoff
    reply /Excessive unknown recipients - possible Open Relay/  mode=backoff
    reply /^421 .* too many errors/                             mode=backoff
    reply /permanently deferred/                                bounce-queue
    reply /too many connections/                                skip-mx
    reply /account over quota/                                  bounce-rcpt
    reply /TS03/               mode=backoff,disable-source-ip,reenable-after=1h
    reply /resources temporarily unavailable/                  defer-queue
</smtp-pattern-list>

<domain *>
    ...
    smtp-pattern-list blockList
</domain>
```

To match on a # (the comment symbol in the configuration file) hexadecimal code may be used. For example, to match the pattern "Sender address deferred by rule #CR-IN-DEF-2":

```
<smtp-pattern-list blockList>
    reply /Sender address deferred by rule \x23CR-IN-DEF-2 /    mode=backoff
</smtp-pattern-list>

<domain *>
    ...
    smtp-pattern-list blockList
</domain>
```

Backslashes (\) and quotes (") will need to be escaped with a backslash. All patterns are case insensitive. If case sensitivity is needed, it can be forced by adding (?-i) like the following example:

```
<smtp-pattern-list blockList>
    reply /(?-i)TS03/    mode=backoff
</smtp-pattern-list>
```

3.3.18 DNS Directives

precached-domains-file
> **Scope:**   global
> **Type:**   string
> **Attributes:** optional
> **Default:**

> Specifies the name of a text file containing a list of domains, one domain per line, that you want PowerMTA to always have current DNS information for in its DNS cache, regardless of whether recipients are in the queue for the domain or not. Typically PowerMTA would look up domain routing information in DNS when recipients are submitted into PowerMTA for a domain (assuming recipients are not currently queued for the domain already). Depending on the DNS infrastructure and how many other domains are in the queue, this may cause some slight delay in the immediate delivery attempts for these recipients. Having DNS routing information precached in PowerMTA for certain key domains before submission minimizes any delay that may take place due to this required DNS resolution.

precached-refresh-interval
> **Scope:**   global
> **Type:**   time
> **Attributes:** optional
> **Default:**   5m

> Defines the time frame in which the DNS records for all pre-cached domains are refreshed (looked back up in DNS). Similar to a DNS record's TTL (time to live), however it manually sets the time interval to refresh, and which overrides the actual records' TTL in DNS. The regular TTL records for pre-cached domains are ignored.

`precached-max-domains`

> **Scope:** global
> **Type:** number
> **Attributes:** optional
> **Default:** 500

Specifies the maximum number of domains that can be pre-cached. This number cannot exceed the number of domains defined in the "precached-domains-file". Defining a very large number will result in lots of frequent DNS queries, so a scalable DNS infrastructure is required.

3.3.19 Other Directives

`auto-qualify-domain`

> **Scope:** source
> **Type:** boolean
> **Attributes:** optional
> **Default:** true

Specifies whether PowerMTA should automatically qualify the domain name in any e-mail addresses received whose domain is not yet fully qualified. For example, if `MAIL FROM:<user@host>` is received and the host running PowerMTA is named `mail.foo.com`, then that address is changed into `user@host.foo.com`. Automatic qualification is done by appending the *domain suffix*, which is also configurable.

`host-id`

> **Scope:** global
> **Type:** number string
> **Attributes:** optional
> **Default:**

When using "add-message-id-header", host-id specifies a per server unique identifier that helps prevent the same message ID from being used on two separate instances of PowerMTA (multiple servers). Only takes numbers as a value.

reroute-to-virtual-mta

> **Scope:** domain
> **Type:** text
> **Attributes:** optional
> **Default:**

Specifies that PowerMTA should reroute messages to the given VirtualMTA. This can be set globally like this example:

```
<virtual-mta vmta1>
    ...
</virtual-mta>

<virtual-mta vmta2>
    ...
</virtual-mta>

<domain aol.com>
    reroute-to-virtual-mta vmta2
</domain>
```

Or, it may also make sense to use this directive on a per VirtualMTA basis:

```
<virtual-mta vmta1>
    ...
    <domain aol.com>
        reroute-to-virtual-mta vmta2
    </domain>
</virtual-mta>

<virtual-mta vmta2>
    ...
</virtual-mta>
```

A reroute to a VirtualMTA pool is not supported. "`reroute-to-vmta`" can be used in place of "`reroute-to-virtual-mta`".

domain-suffix

> **Scope:** global
> **Type:** domain
> **Attributes:** optional
> **Default:** system's domain suffix

Specifies the domain name to append when an e-mail address is received whose domain is not fully qualified.

host-name

> **Scope:** global, virtual-mta
>
> **Type:** domain list
>
> **Attributes:** optional
>
> **Default:** system's host name

This directive specifies a (fully qualified) host name for the local host. If the machine on which PowerMTA is running is known under several names, these should all be listed by adding various `host-name` directives. The first name specified is picked as the "main" name and is used for the SMTP greeting messages, `EHLO`/`HELO` commands, etc.

When specified within a VirtualMTA, this directive overrides the global host name for messages sent through that VirtualMTA.

When used, this directives behaves like relay-domain, meaning that it will accept all messages for the defined domain.

include

> **Scope:** global, source, domain, virtual-mta
>
> **Type:** file name
>
> **Attributes:** optional
>
> **Default:** none

This directive specifies an additional configuration file to process. This can be used, for example, to facilitate maintenance of the configuration files across multiple hosts, by storing those settings which differ from host to host in a separate file and including it from the main (common) file. Wildcards may be used. An include may be used anywhere including in a <source>, <domain>, or <virtual-mta> tag.

```
include /etc/pmta/vmtas.txt
include /etc/pmta/includes/*
```

name-server

> **Scope:** global
>
> **Type:** IP address
>
> **Attributes:** optional
>
> **Default:** system's name servers

This directive specifies the IP address of a name server (DNS server) for PowerMTA to use. If none is entered, PowerMTA uses the name servers configured in the operating system. If more than one name-server is defined, PowerMTA will round robin through all defined name servers in a load balancing fashion.

thread-min-priority, thread-max-priority

> **Scope:** global
>
> **Type:** number
>
> **Attributes:** optional, non reloadable
>
> **Default:** see text

Unix specific directive. Specifies range of process priorities to use for the various threads within PowerMTA. Normally these directives should be left at their defaults, however it may be necessary to weight process priorities differently when PowerMTA shares a host with other server software. `thread-min-priority` specifies the "slower" priority and `thread-max-priority` specifies the "faster" priority.

thread-reuse

> **Scope:** global
>
> **Type:** boolean
>
> **Attributes:** optional, non reloadable
>
> **Default:** true

Normally, once a thread is finished with its work (like, for example, handling a connection), PowerMTA retains it for later reuse. Setting this directive to `false` causes PowerMTA to discard threads as soon as they are no longer needed, avoiding certain external problems such as slowing down the `top` command on Linux. However, this costs some performance and also prevents PowerMTA from working around a bug in glibc versions prior to 2.2.2 that severely reduces the amount of virtual memory available, possibly causing PowerMTA to crash if the queue grows very large.

`mailmerge-expands-undefined-variables`

     Scope: global

     Type: boolean

     Attributes: optional, non reloadable

     Default: true

     When set to "false", mail merge inserts like "[foo]" remain as "[foo]" in the message if the "foo" variable isn't defined, rather than be expanded to the empty string.

`pmc-acct-min-free-space`

     Scope: global

     Type: {n{B|K|M|G|T}|unlimited}

     Attributes: optional, reloadable

     Default: 5M

     Controls when to start deleting old CSV files written for dbloader.

141

4. PowerMTA Management

Although PowerMTA has been designed for being as low maintenance as possible, it does include various tools for management and troubleshooting. This chapter is intended as a guide to daily operations with PowerMTA, for those who wish to actively manage and monitor, as well as to troubleshooting the most commonly encountered problems.

4.1 Management

4.1.1 Monitoring

There is a number of ways in which you can monitor PowerMTA. It has a web-based monitor built in, so you can simply point your web browser to the monitor's TCP port on PowerMTA's machine to view the mailer's current status and display information about its queue. This comes particularly handy when monitoring over longer periods of time. The web monitor is described in detail in Chapter 5.

The same information from the web monitor also available through the `pmta` command line tool. In addition to displaying the command results in formatted plain text, `pmta` can also output these in XML or as DOM-Style variable lists, making it relatively easy to post-process and to integrate PowerMTA in existing management infrastructures. The `pmta` command line tool and the commands it accepts are described in Chapter 6.

4.1.2 Logging and Accounting

For greatest flexibility and best performance, PowerMTA maintains its own logging file. The operating system's log (i.e., the Event Log on Windows or the "syslog" on Unix) is only used for recording information on startup errors and similar situations where it may not be possible for PowerMTA to write to its own logging file.

The logging file is used for recording various information about PowerMTA, such as startup and shutdown messages, any errors found during execution, connection traces, etc. In general terms, the log file is the place where the mailer writes any error messages, and you should look there first when any problems arise.

In addition to the logging file, PowerMTA records detailed information on e-mails processed in a separate *accounting file*. Storing accounting data separate from other, more general, logging data allows for a more efficient accounting file format while retaining the logging file in plain text. It also makes it easier to send feedback about problems with PowerMTA while keeping the accounting data undisclosed. Section 11 describes the

format of the accounting files, as well as a sample accounting statistics application provided with PowerMTA.

Both logging and accounting files are *rotated* automatically, that is, a new file is started and the older ones are kept for a number of days. The parameters for the rotation, including the number of (historical) files to keep can are described in [Section 3.2.6](#) and in [Section 3.2.7](#).

4.1.3 PowerMTA Startup and Shutdown

On Windows systems, PowerMTA runs as a service and as such is started up automatically by the operating system after booting. It also can be started and stopped both via the Control Panel and by using the `net start pmta` and `net stop pmta` commands from a Command Prompt window. Currently, PowerMTA must run under the `SYSTEM` account.

On Unix systems, PowerMTA runs as a daemon, i.e., a background process. A standard startup/shutdown script is provided in the system's `init.d` directory.

On either platform, two processes are started: `pmtad`, which is PowerMTA itself, and `pmtawatch`, which is PowerMTA's watchdog process. As the name suggests, `pmtawatch` watches over PowerMTA and re-starts it in case of a (PowerMTA) crash.

4.2 Troubleshooting

The following sections guide through troubleshooting the most commonly encountered problems.

4.2.1 Troubleshooting Startup Problems

Startup problems typically occur right after installing PowerMTA or after having changed its configuration. If PowerMTA won't start up, you should look in the logging file for an error message. On Windows, the logging file is named `log.txt` and is located in the `log` subdirectory of the folder where PowerMTA was installed. On Unix systems, the logging file is located on `/var/log/pmta/log`.

If no logging file is present or if it does not seem to contain an applicable error message, you can try running PowerMTA on the command line. On Windows, open a Command Prompt window and change to PowerMTA's `bin` subdirectory. On Unix, ensure that `/usr/sbin` is in your path. Then start PowerMTA by using the command

```
pmtad --debug
```

That will start up PowerMTA and will have it write any error messages to the screen, rather than to the logging file. Most probably, the startup will fail displaying an error message, like in the example below:

```
# /usr/sbin/pmtad --debug
2000-07-03 18:05:37 Startup error: No logging file specified
```

You can then fix the problem and re-start it. If PowerMTA does start, you can shut it down by simply pressing Control-C once and waiting for it to stop. On Windows, pressing Control-C repeatedly causes it to do an "emergency exit". On Unix, you can achieve a similar effect by pressing Control-\ (thus sending it the SIGQUIT signal). Use the emergency exits only if absolutely necessary (e.g., if the mailer won't shut down after waiting for several minutes), since it may cause PowerMTA to loose track of which recipients have already been handled, causing duplicate deliveries.

A common cause of startup problems when PowerMTA is first installed is TCP port conflicts. Applications that listen for (i.e., accept) connections on a TCP port need exclusive access to the port, so if you are running other SMTP server software on the same machine, either PowerMTA or the other software will fail to obtain access to port number 25 (the standard port for the SMTP protocol). The same might also happen with the TCP port used for the built-in web monitor (port 8080 by default). In the case of the the web monitor, since 8080 is just a local convention, you can resolve the conflict by changing the configuration (see Section 3.2.5) and specifying the new port in your web browser.

The SMTP port conflict may be more difficult to solve. If you are *not* using PowerMTA for handling inbound traffic, i.e., you are just feeding PowerMTA through SMTP or the submission APIs and inbound traffic is handled by some other software, you can configure PowerMTA to use an alternative SMTP port (see Section 3.2.4) and change your feeding software to use that port. You can normally pick any free TCP port above 1024; port numbers 2500 and 2525 are popular choices.

If, on the other hand, both PowerMTA and the other SMTP server software are to receive inbound traffic, you will need to give each of them a different IP address. Normally, applications listening for (i.e., accepting) connections on a TCP port will do so for all locally available IP addresses, so *both* PowerMTA and the other SMTP software need to be configured to use a specific IP address only (see Section 3.2.4 for PowerMTA's configuration). Naturally, you will first need to allocate a new IP address and configure your operating system to respond to it.

4.2.2 Troubleshooting E-mail Delivery Problems

E-mail delivery problems constitute by far the most common type of problem encountered. It is quite normal for some e-mail not to be delivered immediately and for

144

the delivery to have to be retried. The causes of such problems are often invalid e-mail addresses, network or server failures or misconfigured software at the recipient's site.

If you notice e-mail building up in the queue for a specific domain, the first step is to check the (DNS-based) routing information for the domain. If that information is available and seemingly correct, the next step would be to check whether any of the listed mailers is reachable. PowerMTA's `pmta` command line tool has a `resolve` command that allows you to do all of that in a single step:

```
$ pmta resolve --connect hotmail.com
Querying 193.175.161.130 over UDP about MX(15) hotmail.com

status = StatusOk (no error)
pref- host name-------------------- IP addresses/resolution status----
--------
    10 mc2.law5.hotmail.com          216.32.243.135
    10 mc4.law5.hotmail.com          216.33.151.136
    10 mc5.law5.hotmail.com          216.32.243.136
    10 mail.hotmail.com              216.33.151.135

Connecting to mc2.law5.hotmail.com (216.32.243.135)... ok
>>> 220-HotMail (NO UCE) ESMTP server ready at Tue Jul 04 08:02:57 2000
>>> 220 ESMTP spoken here
<<< ehlo test
>>> 250-hotmail.com Hello
>>> 250 SIZE 1048576
<<< quit
>>> 221 Service closing transmission channel
```

In the example above, routing information for the domain `hotmail.com` could be resolved. It listed four mail hosts, all with preference level 10, each with its own IP address. The `--connect` option specified that `pmta` attempts connecting to the mailers (which it did, to "mc2").

When you enter this command during troubleshooting, either part of the command may fail. You may get an error during the DNS resolution part, like, for example "No such domain name" for a domain that is not registered in the DNS. Similarly, if the routing information is available but if the mailers themselves are not reachable, the connection attempt fails.

Simply connecting to the mailers does not preclude other problems delivering e-mail, for the problem may be related to the specific messages queued, or the mailers may be temporarily rejecting e-mail (e.g., returning a 4XX code to SMTP commands). To verify if this is happening, use the `pmta trace` command. As described in [Section 6.3.21](#), the `trace` command instructs PowerMTA to retry e-mail delivery to the domain immediately and log the connection. This way, you can see (in the logging file) exactly what happens when delivery is attempted.

The `pmta trace` command is useful for quickly checking on a domain. If you should need to trace connections to a domain over a longer period of time, enable logging for it in the configuration file by adding a `<domain>` entry, like in in the example below:

```
<domain hotmail.com>
    log-connections yes
    log-commands yes
</domain>
```

Please remember to add such domain-specific entries *before* the `<domain *>` entry present in most configuration files and to reload the configuration by using `pmta reload`.

4.2.3 Troubleshooting DNS Name Resolution Problems

PowerMTA requires a recursive name service, i.e., a name server that accepts queries and performs the full DNS resolution on its behalf.

Some name servers fail to do the recursive lookup and return non-recursive (referral-only) responses instead. These responses don't contain the information sought by PowerMTA, so it discards them and retries the query in the hope that whatever caused them is temporary, or that some other server will provide the full response. When such responses are received a number of times from the same name server, PowerMTA also writes a warning to its logging file.

Generally speaking, such non-recursive responses may have various causes. However, some DNS server software versions are known to send them when configured to refuse recursive service, or when its limit of concurrent recursive queries has been reached.

Non-recursive responses waste time and may have an impact on performance. If you find a name server refusing recursive queries too often, check the name server configuration for any quotas or access control lists. If you do not control the name server, you should consider no longer using it for PowerMTA and setting up your own.

4.2.4 Troubleshooting E-mail Reception/SMTP Feeding Problems

Problems receiving e-mail are quite rare and happen most commonly when writing custom applications to feed PowerMTA. The most common mistake when writing such applications is to terminate lines with just a line feed, rather than a carriage return and line feed pair as mandated by the e-mail standards.

The simplest way to debug incoming SMTP connections into PowerMTA is to enable debugging for a specific source IP address. You do so by adding a `<source>` entry in the configuration file, like in the example below,

```
<source 127.0.0.1>
    log-connections yes
    log-commands yes
</source>
```

and reload the configuration by using the `pmta reload` command. This will cause PowerMTA to trace incoming connections to the logging file, which helps determine in what phase of the transaction (for example, in a `RCPT` command) the problem is occurring. If you suspect the problem to lay in non-printable characters or in the message's body, you can additionally enable `log-data`, which prints all data sent or received through the connection along with each byte's hexadecimal code. For more information on the logging options available, please see Section 3.2.6.

4.2.5 In Case of a Crash

If PowerMTA crashes, its `pmtawatch` watchdog program will attempt to re-start it and you should get an e-mail informing you about the crash. Please forward this e-mail to us at `support@port25.com`, so that we can look into the problem. Feel free to include any information you may find relevant, such as recent changes to the configuration or workload, etc.

On Unix systems, the core dump will be stored, compressed if possible, in PowerMTA's subdirectory on the `/opt` filesystem.

4.2.6 If All Else Fails

Check out http://www.port25.com/support/ for technical support options from Port25. Evaluators and customers with a support agreement may also get support via e-mail at support@port25.com or over the phone at +1.410.203.2323.

5. Web-Based Monitoring

5.1 Overview

PowerMTA has a web-based monitoring console built in that provides real time display of PowerMTA's status and allows you to view information on the current message queue based on various criteria. The idea is to continue to develop this monitor into a full-blown management console. This chapter describes the functionality implemented to date.

To access the web monitor, point your web browser to the monitor's TCP port on the machine you installed PowerMTA. For example, if you are reading this document online and from the same machine running PowerMTA, you would use the URL http://127.0.0.1:8080. This also assumes that you did not change the default TCP port in the configuration. See Section 3.2.5 for more information on the web monitor's configuration.

When you try the URL, you may get a message saying that the access is forbidden. This is because, to protect your privacy, access to the web monitor is denied unless authorized in the configuration. Authorization is done on a per source IP address basis, so you need to enter the IP address of the host running your web browser into the configuration.

148

5.2 Home Display

The initial "Home" page displays basic information about your installation. In addition to this information, there are four useful links:

- Edit Configuration – This allows you to view and edit the configuration file from within the web monitor window. After saving changes, a "pmta reload" will not be required as it it done automatically. If the change requires a restart of the service, that will have to be done manually.

- Show/enter License Key – This links to a form where you can view/add/change the license.

- Run command – This link allows you to run "pmta" commands as if you were logged in a the command/shell prompt of the PowerMTA server.

- User's Guide – Opens UserGuide.pdf in a new window for ease of looking up PowerMTA information.

- Port25 Support – Starts an email to support @port25.com using your default email program.

From the home page, you have the option of changing to other displays:

- *Status*, the status monitor page, which displays various information about PowerMTA's status, updated every few seconds;
- *Queues*, which presents you with various options for listing queues to which e-mail is queued.
- *Domains*, which presents you with various options for listing domains to which e-mail is queued.
- *VirtualMTAs*, which displays a summary of the VirtualMTAs in use;
- *Jobs*, which presents you with summary information on any jobs (groups of messages) queued.
- *Logs*, allos for the viewing and downloading of PowerMTA's various log files.

The following sections describe these pages in more detail.

5.3 Status Monitor Display

The status monitor displays data from various components within PowerMTA.

Inbound Traffic	rcpts	msgs	kbytes
total	368,736,950	9,147,779	138,466.9
last hour	2,583,612	63,947	967.9
top/hour	2,589,256	64,557	977.3
last minute	43,000	1,053	16.0
top/minute	44,000	1,159	17.5

Outbound Traffic	rcpts	msgs	kbytes
total	368,957,414	368,957,414	122,333,637.9
last hour	2,583,483	2,583,483	855,222.0
top/hour	2,768,752	2,768,752	918,627.0
last minute	42,957	42,957	14,151.2
top/minute	261,385	261,385	86,752.1

Connections	in	out
active	1	0
top	1	500
max	150	500

Domain Names	
cached	0
pending	0

Spool	
files	4,221
directories	2
initialization	100%

Queues	rcpts	domains	kbytes
SMTP	5,862	293	88.8
discard	0	0	0.0
file	0	0	0.0
pipe	0	0	0.0

Status	running	Started on	2015-07-23 14:28:47	Uptime	5 23:13:08

The "Total" counters show the total data received and sent since the mailer was started. Similarly, the "Last Hour" and "Last Minute" counters show the total data received and sent for the respective time intervals. Note that this has been changed from the earlier beta versions which extrapolated the data up to the time increment if the mailer had not been running for a full hour or full minute.

The "Top/Hour" and "Top/Minute" lines show the highest throughput observed per hour and minute. The various counters are measured independently of each other, i.e., the top number of inbound recipients may or may not have happended at the same time as the top number of inbound kbytes.

The "Connections" section shows the number of currently active connections, the top number of simultaneous connections made since last restart, and the maximum number of inbound and outbound simultaneous connections available given your license activation key and configured limits.

The "Domain Names" section shows the number of DNS entries currently in PowerMTA's internal cache, as well as the number of DNS queries pending, i.e., currently taking place or scheduled and waiting to take place.

The "Spool Files" section shows the number of spool files currently being used for storing queued messages. And, since the spool management code recycles files for efficiency instead of having to recreate as needed, this section also tracks the number of files that have been recycled and are waiting to be reused.

The "Spool Initialization" section shows whether the initialization is in progress or has been completed. This initialization takes place at startup and consists of reading the message files in the spool and loading queue information into memory.

The "Queues" section provides top level counts for each of the delivery methods available: SMTP or through the pipe delivery API. The counts include the total number of recipients currently in the queue, total number of destination domains for these recipients, and the total amount of data in the mail queues to be delivered, for each method.

The default refresh interval is five seconds. You can select a different refresh interval by specifying a `refresh` parameter in the URL. For example, to specify a refresh interval of five minutes (300 seconds), you could use an URL like http://127.0.0.1:8080/status?refresh=300. The refresh interval is independent of the internal throughput sampling interval, and as such has no effect on the display's accuracy. It thus normally makes little sense to specify refresh intervals *below* the default. Specifying refresh intervals above the default interrupts PowerMTA less often, which may have a positive impact on performance.

This page can also be retrieved in XML format for processing by a program. You specify the XML format by adding `format=xml` to the URL. For example, you could use the `wget` utility to retrieve it:

```
$ wget -O- -q http://127.0.0.1:8080/status?format=xml
<rsp><data><mta><product><name>PowerMTA</name><version>2.0r1</version>
<buildDate>Jun 30 2003 19:15:23</buildDate></product><os><name>Linux</name>
<version>2.4.18-10</version><build>Red Hat Linux release 7.3 (Valhalla)</build>
</os><cpu><type>i686</type><count>1</count></cpu><ram><real>513800</real></ram>
<fullHostName>hazmat.port25.com</fullHostName><userString></userString></mta>
<status><timeNow>1031828706</timeNow><startupTime>1031815851</startupTime>
<shuttingDown>0</shuttingDown><traffic><total><out><rcp>1</rcp><msg>1</msg>
<kb>1.7</kb></out><in><rcp>1</rcp><msg>1</msg><kb>1.6</kb></in></total>
<lastHr><out><rcp>1</rcp><msg>1</msg><kb>1.7</kb></out><in><rcp>1</rcp>
<msg>1</msg><kb>1.6</kb></in></lastHr><lastMin><out><rcp>0</rcp><msg>0</msg>
<kb>0.0</kb></out><in><rcp>0</rcp><msg>0</msg><kb>0.0</kb></in></lastMin>
```

```
<topPerHr><out><rcp>1</rcp><msg>1</msg><kb>1.7</kb></out><in><rcp>1</rcp>
<msg>1</msg><kb>1.6</kb></in></topPerHr><topPerMin><out><rcp>1</rcp>
<msg>1</msg><kb>1.7</kb></out><in><rcp>1</rcp><msg>1</msg><kb>1.6</kb></in>
</topPerMin></traffic><conn><smtpIn><cur>0</cur><max>30</max><top>0</top>
</smtpIn><smtpOut><cur>0</cur><max>800</max><top>1</top></smtpOut></conn>
<resolver><namesCached>5</namesCached><queriesPending>0</queriesPending>
</resolver><queue><smtp><rcp>0</rcp><dom>0</dom><kb>0.0</kb></smtp></queue>
<spool><initPct>100</initPct><dirs>1</dirs><files><inUse>0</inUse>
<recycled>1</recycled><total>1</total></files></spool></status></data></rsp>
```

5.4 Queues Display

Similar to the "Top Domains" page, the "Top Queues" page provides a listing of the top ten queues ranked by number of recipients.

For greatest flexibility and control, PowerMTA groups messages for a particular domain for a particular VirtualMTA in its own separate queue. For example, mail destined for `msn.com` and handled by VirtualMTA `mta5` would be in a different queue than mail destined for `msn.com` and handled by VirtualMTA `mta10`.

Whereas the "Top Domains" page provides a global view with the top ten domains across all queues, the "Top Queues" page provides a listing and counts for specific queues.

port25 solutions, inc.

PowerMTA™ 4.5b6
mail1.port25.com

| Home | Status | Queues | Domains | Virtual MTAs | Jobs | Logs |

Top 10 queues by number of recipients

Customize View Help

Name	#Rcpt	KBytes	#Conn	Paused	Mode	Last Error
yAhOO.ES/vmta1 ⚠	52	0.8	0	no	normal	2015-07-23 14:29:42 message rate limit reached (based on queue's max-msg-rate in configuration) (23x)
aOL.Com/vmta1 ⚠	45	0.7	0	no	normal	2015-07-23 14:29:32 message rate limit reached (based on queue's max-msg-rate in configuration) (7x)
wAnadOo.FR/vmta1	43	0.7	0	no	normal	
siNa.cOm/vmta1 ⚠	39	0.6	0	no	normal	2015-07-23 14:29:11 message rate limit reached (based on queue's max-msg-rate in configuration) (3x)
YAhOo.fR/vmta1 ⚠	39	0.6	0	no	normal	2015-07-23 14:29:42 message rate limit reached (based on queue's max-msg-rate in configuration) (25x)

The individual queue pages also include data on the last 40 recipient events.

port25 solutions, inc.

| Home | Status | Queues | Domains | Virtual MTAs | Jobs | Logs |

Help

Queue Detail

name	yAhOO.ES/vmta1
recipients	56
kbytes	0.8
active connections	0
paused	no
mode	normal
backoff reason	
next retry	2015-07-29 13:53:29

Last Errors

time	error
2015-07-23 14:29:42	message rate limit reached (based on queue's max-msg-rate in configuration) (23x)
2015-07-23 14:29:28	message rate limit reached (based on queue's max-msg-rate in configuration) (14x)
2015-07-23 14:29:13	message rate limit reached (based on queue's max-msg-rate in configuration) (8x)
2015-07-23 14:29:00	message rate limit reached (based on queue's max-msg-rate in configuration) (3x)

Recipient Events   (Deliveries: 40, Bounces: 7, Deferrals: 3)

Hide

Event	Time	Source IP	Address	Job ID	Destination MTA	Additional Information
delivered	2015-07-29 13:52:29	127.0.0.1	BB@yAHoO.eS		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
deferred	2015-07-29 13:52:28	127.0.0.1	A@yAhoO.es		[127.0.0.1] (127.0.0.1)	smtp;452 4.2.2 Disk quota exceeded
delivered	2015-07-29 13:52:27	127.0.0.1	IA@yaHoo.eS		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
delivered	2015-07-29 13:52:27	127.0.0.1	UA@YaHOo.eS		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
delivered	2015-07-29 13:52:27	127.0.0.1	C@YAhoO.ES		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
delivered	2015-07-29 13:52:27	127.0.0.1	BA@YAHOO.ES		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
delivered	2015-07-29 13:52:27	127.0.0.1	BR@yAhoo.eS		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
delivered	2015-07-29 13:52:27	127.0.0.1	W@YaHoO.eS		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
delivered	2015-07-29 13:52:26	127.0.0.1	A@YaHoo.eS		[127.0.0.1] (127.0.0.1)	smtp;250 DATA ok
deferred	2015-07-29 13:52:26	127.0.0.1	BD@YahOO.eS		[127.0.0.1] (127.0.0.1)	smtp;452 4.2.1 mailbox temporarily disabled:

This information is also available in the output of the pmta show queues command.

5.5 Queues "View Options"

The "Queues" page consists of a Queues "view options" form which after submitting, generates an output similar in format to the "Top Queues" page, but for the specific queue(s) specifed in conjunction with relevant filters.

The filters include:

- connected queues only, unconnected queues only, or both
- queues in normal mode, backoff mode, or any mode
- the desired number of queues to list

One can then choose how to sort the output, either by queue name, number of recipients, or queue size.

5.6 Domains Display

The "Top Domains" page provides a listing of the top ten domains in the queue ranked by number of recipients.

Name	#Rcpt	KBytes	#Conn	Last Error
yAhOO.ES	73	1.1	0	2015-07-23 14:29:42 message rate limit reached (based on queue's max-msg-rate in configuration) (23x)
aOL.Com	70	1.1	0	2015-07-23 14:29:32 message rate limit reached (based on queue's max-msg-rate in configuration) (7x)
YAhOo.fR	67	1.0	0	2015-07-23 14:29:42 message rate limit reached (based on queue's max-msg-rate in configuration) (25x)
ALICE.it	59	0.9	0	2015-07-23 14:29:25 message rate limit reached (based on queue's max-msg-rate in configuration) (6x)
wAnadOo.FR	56	0.8	0	

© 2016, Port25 Solutions, Inc. This document contains confidential and proprietary information belonging to Port25 Solutions, Inc. Not for release or distribution without permission. (v.2016-12-06-01)

For each domain listed, the number of the number of recipients, amount of data (kbytes), the number of open connections and the last error occurred while delivering to the domain are displayed.

Because this page is relatively resource intensive to build, the default refresh interval is two minutes. You can select a different refresh interval by specifying a `refresh` parameter in the URL. Additionally, you can also request that more domains be displayed by entering a `maxItems` parameter. For example, to specify a refresh interval of five minutes (300 seconds) and twenty top domains, you could use an URL like http://127.0.0.1:8080/topdomains?refresh=300&maxItems=20.

5.7 Domains "View Options"

The "Domains" page consists of a Domains "view options" form which after submitting, generates an output similar in format to the "Top Domains" page, but for the specific domain(s) and based on the other parameters specifed.

View options			
Domain name	*		
Virtual MTA	*		
Sort by	○ name ◉ # of recipients ○ queue size		
Connections	○ connected domains only ○ unconnected domains only ◉ both		
Display at most	10	domains	
Refresh every	120	seconds (if omitted or zero, don't refresh)	

Display

5.8 Common "Last Error"s

Included below is a list of the most common "Last Error" messages, with an explanation about what may cause them. On Windows, system status codes may be prefixed with `WSA`, but indicate generally the same condition.

ETIMEDOUT in connection to *domain* (*IP*)
> This error indicates that the connection failed because, while connected, the remote host did not properly respond within a period of time. It possibly indicates that the remote host is overwhelmed or that there are connectivity problems. PowerMTA's connection timeout parameter is 10 minutes, and is not configurable.

ECONNREFUSED connecting to *domain* (*IP*)
> This error indicates that no connection could be made because the remote host actively refused it. This usually results when the remote mailer software is temporarily down or inactive, or if the remote site is actively refusing connections (usually with a packet filter) from your company and/or IP address.

156

EDISCON in connection to *domain* (*IP*)

    This error indicates that the remote host has unexpectedly closed the connection. It may result from the remote mail server being taken down, crashing, or simply closing the connection, possibly as a "last resort" kind of response to message contents it could not process.

StatusDnsQueryFailed resolving domain

    This error indicates that the DNS query for the domain could not be completed. This could be due to more than one reason, specifically timeouts when PowerMTA attempts to talk to your local DNS server, but more likely, timeouts in your local DNS server itself when attempting to talk to other DNS servers. Although these errors can and will be seen for some domains that appear to not have valid MX or A records, the fact that the actual DNS queries timeout before receiving any verification of this precludes PowerMTA from making any assumptions and thus, for example, bouncing mail to these domains immediately.

EHLO failed

    These are errors that are returned by the remote mail server during the initial SMTP handshake (at the `HELO` or `EHLO` SMTP commands). Usually the text included in the error messages is self-explanatory. Here are a few examples:

```
421 domain Service not available - too busy
421 domain Sorry, you are not authorized to make this connection.
421 domain temporarily not accepting mail
452 domain Too busy, please try later.
```

In order to avoid filling its limited space with irrelevant errors, PowerMTA is quite selective about which events it records.

If you see domains in the queue with no errors specified, along with no active connections, this means that the error(s) occurred during the last attempt is not among those events that PowerMTA records. For these domains, we recommend running the `trace` command from the command line interface, as detailed Section 6.3.21, or enable one or more of the various types of logging available as detailed in Section 3.2.6.

As we continue to improve upon this feature, more events and error conditions will be recorded.

5.9 VirtualMTAs Display

port25
solutions, inc.

| Home | Status | Queues | Domains | Virtual MTAs | Jobs | Logs |

Virtual MTAs with messages in the queue

Help

name	Domains	#Rcpt	% Total	KBytes	#Conn
vmta1	291	4,140	70%	62.7	0
vmta21	42	45	0%	0.7	0
vmta28	37	42	0%	0.6	0
vmta39	39	40	0%	0.6	0
vmta2	35	40	0%	0.6	0
vmta27	36	38	0%	0.6	0
vmta13	36	37	0%	0.6	0
vmta16	34	37	0%	0.6	0
vmta23	37	37	0%	0.6	0
vmta50	34	37	0%	0.6	0
vmta4	36	37	0%	0.6	0
vmta35	33	36	0%	0.5	0
vmta5	34	36	0%	0.5	0
vmta41	35	35	0%	0.5	0
vmta42	33	35	0%	0.5	0
vmta49	31	35	0%	0.5	0
vmta30	31	34	0%	0.5	0
vmta29	33	34	0%	0.5	0
vmta46	29	34	0%	0.5	0
vmta14	32	33	0%	0.5	0

The VirtualMTAs page provides a listing of the VirtualMTAs selected that currently have messages in the queue. The VirtualMTAs are listed sorted by number of recipients.

Each VirtualMTA listed also links to a top domains page that subsequently lists the top 10 domains in the queue for the specific VirtualMTA, ranked by number of recipients in the queue. Also included on this page is the volume of data in the queue, number of current connections open to the domain, and the last error received for the domain, again for the specific VirtualMTA.

5.10 Jobs Display

For each job listed, the number of the number of recipients and amount of data (kbytes) are displayed.

As with most pages served by PowerMTA, you can specify a refresh interval in seconds by adding it to the URL. For example, to specify a refresh interval of five minutes (300 seconds), you could use an URL like http://127.0.0.1:8080/jobs?refresh=300.

Like with the status display, this page can also be retrieved in XML format for processing by a program, by adding `format=xml` to the URL.

5.11 Logs Display

The Logs page displays the various log and accounting files currently available in PowerMTA. By choosing the download link, these files can be downloaded directly to your local machine for viewing. In addition, these files can be accessed programmatically via the URL format: http://127.0.0.1:8080/getFile?file=log.txt where "log.txt" is the name of the file you want to download.

port25 solutions, inc.

PowerMTA™ 4.5b6
mail1.port25.com

| Home | Status | Queues | Domains | Virtual MTAs | Jobs | Logs |

Help

Logging files

file	date	size	
pmtahttp.log	2015-07-29 13:50:02	180K	download
pmta.log	2015-07-29 13:49:20	84K	download
pmta.log.1	2015-07-29 00:00:00	145K	download
pmtahttp.log.1	2015-07-29 00:00:00	392	download
pmtahttp.log.2	2015-07-28 00:00:00	85K	download
pmta.log.2	2015-07-28 00:00:00	145K	download
pmtahttp.log.3	2015-07-27 00:00:00	46	download
pmta.log.3	2015-07-27 00:00:00	145K	download
pmtahttp.log.4	2015-07-26 00:00:00	46	download
pmta.log.4	2015-07-26 00:00:00	145K	download
pmtahttp.log.5	2015-07-25 00:00:00	1K	download
pmta.log.5	2015-07-25 00:00:00	145K	download
pmtahttp.log.6	2015-07-24 00:00:00	46	download
pmta.log.6	2015-07-24 00:00:00	175K	download
pmtasnmp.log	2015-06-06 13:18:22	228	download
pmtasnmp.log.2	2015-06-05 09:27:10	549	download
pmtasnmp.log.1	2015-06-05 09:27:10	67	download
pmtasnmp.log.3	2015-06-03 18:08:07	570	download
pmtasnmp.log.4	2015-06-02 10:59:19	684	download
pmtasnmp.log.5	2015-06-01 16:36:35	67	download
pmtasnmp.log.6	2015-06-01 16:36:35	2K	download

Accounting files

file	date	size	
acct-2015-07-29-0041.csv	2015-07-29 13:50:17	226M	download
acct-2015-07-29-0040.csv	2015-07-29 13:32:23	250M	download
acct-2015-07-29-0039.csv	2015-07-29 13:12:35	250M	download
acct-2015-07-29-0038.csv	2015-07-29 12:52:46	250M	download
acct-2015-07-29-0037.csv	2015-07-29 12:32:58	250M	download

5.12 Listing of local IPs

To get a list of the IPs currently configured on your machine, use the following URL in your web browser: http://127.0.0.1:8080/getlocalips

```
- <rsp>
  - <localIps>
      <ip>192.168.174.1</ip>
      <ip>169.254.127.48</ip>
      <ip>169.254.121.237</ip>
      <ip>192.168.88.1</ip>
      <ip>127.0.0.1</ip>
  </localIps>
  <succ />
</rsp>
```
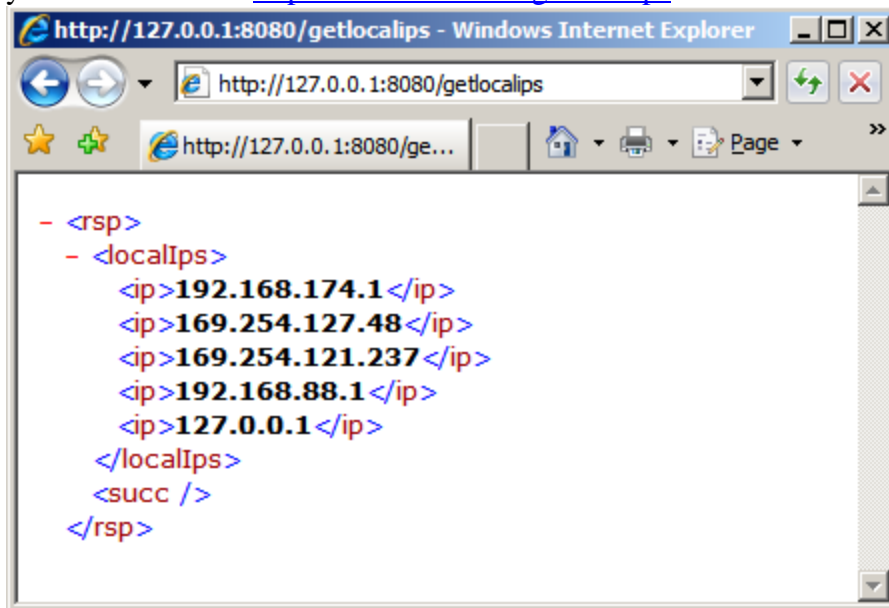
5.13 Listing of emails in queue

To get a listing of the messages that are currently in the queue, use a URL similar to the following in your web browser:

http://127.0.0.1:8080/listqueue?format=xml&queue=domain.com/VirtualMTA&jobid=12
http://127.0.0.1:8080/listqueue?format=xml&queue=yahoo.com/*&jobid=12&envid=3
http://127.0.0.1:8080/listqueue?format=xml&queue=*/vmta12&rcpt=test@port25.com
http://127.0.0.1:8080/listqueue?format=xml&queue=*/*&orig=test@port25.com

Use of jobid is optional. Replace domain.com with the domain for which you are looking. Also replace VirtualMTA with the name of the VirtualMTA in question. Also, use * for all domains or all VirtualMTAs.
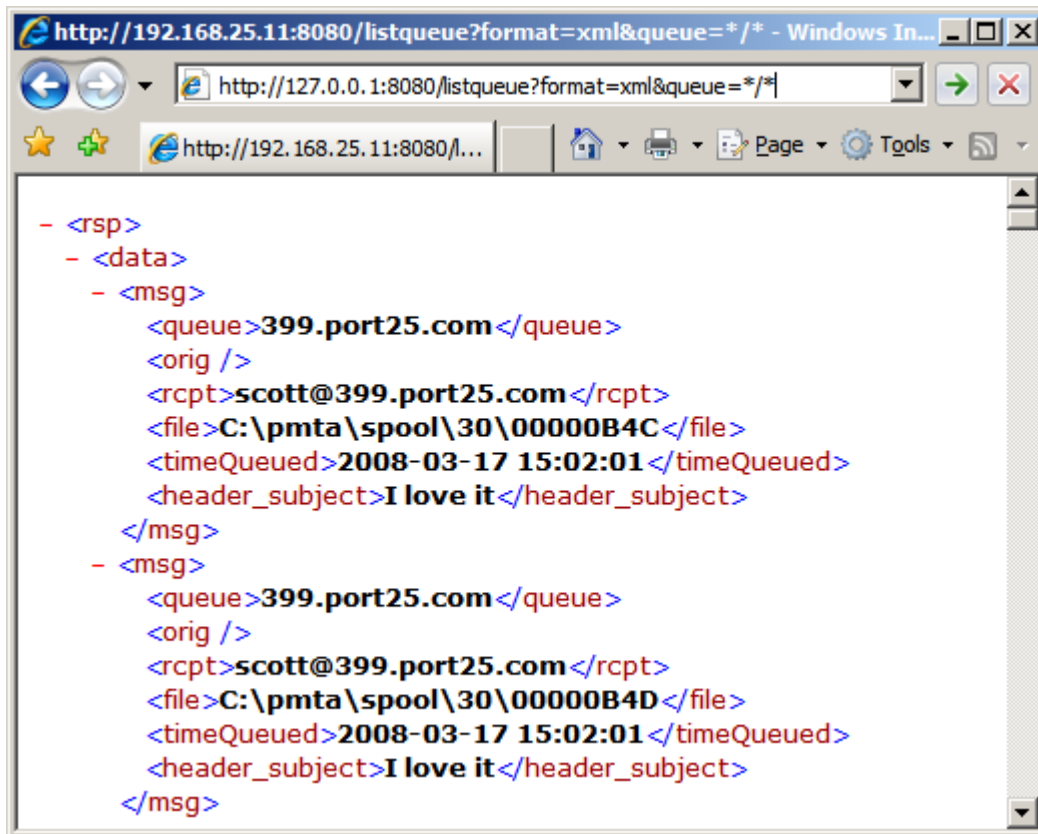
6. Command Line Tool

6.1 Overview

PowerMTA comes with a command line tool that allows you to execute commands, display various information and also generally aids in investigating delivery problems. On Windows, this tool can be found under `bin\pmta.exe` in the folder where PowerMTA was installed. On Unix, it is installed as `/usr/sbin/pmta`. At the command line, you can get a quick listing of the commands available by invoking `pmta --help`:

```
PowerMTA(TM) v4.5 command tool
Copyright(c) 1999-2015, Port25 Solutions, Inc.  All Rights Reserved.

Usage: pmta [options] command

where 'options' are:
    --help              display this message
    --xml               display result in XML
    --dom               display result as DOM-like "variables"

and where 'command' is one of:
    clear dnscache domainname
    check mfrom [--tcp] [--dumppackets] mailfrom ip
    delete [--dsn] [--accounting] [--queue=domain[/vmta]] [--orig=addr] [--rcpt=addr] [--jobId=id] [--envId=id] [--
older-than=time-interval]
    deregister [--user=name] [--local-only] [--retain-unloaded]
    disable source [--reenable-after=interval] ip domain[/vmta]
    enable source ip domain[/vmta]
    list [--queue=domain[/vmta]] [--orig=addr] [--rcpt=addr] [--jobId=id] [--envId=id] [--maxitems=n] [--pause] [--
priority] [--schedule]
    pause queue domain[/vmta]
    pause job jobid
    register [--user=name] [--label=name] [--webmon-ip=ip] [--webmon-port=number] [--pmtamc-port=number] [--reuse-
label] pmtamc-hostname
    reload
    reset counters
    reset status
    resolve [--tcp] [--connect] [--dumppackets] [--interactive] [--source=[host,]ip] domainname
    resume queue domain[/vmta]
    resume job jobid
    rotate acct [file]
    rotate log
    schedule [--retry-recipients] domain[/vmta]
    set priority [--queue=domain[/vmta]] [--orig=addr] [--rcpt=addr] [--jobId=id] [--envId=id] number
    set queue --mode={normal|backoff} domain[/vmta]
    show disabled sources [domain[/vmta]]
    show domains [--vmta=name] [--connected={yes|no}] [--maxitems=n] [--errors] [--sort={name|rcpt|size}] [name]
    show jobs [--maxitems=n]
    show license
    show precache
    show queues [--connected={yes|no}] [--paused={yes|no}] [--mode={normal|backoff}] [--maxitems=n] [--errors] [--
sort={name|rcpt|size}] [--no-rcpt-events] [domain[/vmta]]
    show registration
    show status
    show settings domain[/vmta]
    show topdomains [--vmta=name] [--connected={yes|no}] [--maxitems=n] [--errors]
    show topqueues [--connected={yes|no}] [--paused={yes|no}] [--mode={normal|backoff}] [--maxitems=n] [--errors] [-
-no-rcpt-events] [domain[/vmta]]
    show version
    show vmtas [--maxitems=n]
```

```
trace [--log-data] [--log-resolution] [--to-log] [--retry-recipients] [--source-ip=ip] domain[/vmta]
```

Note: On Windows, you can also use the "/" character instead of "--" (but "--" works on all platforms).

6.2 Command Output Formats

pmta supports three output formats:

- formatted plain text, intended for human consumption, i.e, for the system or services administrator executing PowerMTA commands. It is the default format if no options are passed to pmta.
- XML tags, intended for XML-enabled applications. It is selected by specifying the --xml option to pmta;
- DOM-style variable listing, intended for non XML-capable applications and for quick Perl or shell scripts. It is selected by specifying the --dom option to pmta.

6.2.1 Sample Command Outputs

Included below are three sample command outputs along with the commands and options used to generate each. For comparison purposes, all three are based on the show status command.

Formatted Plain Text (`pmta show status`):

```
        PowerMTA v4.5 status on example.port25.com on 2015-09-01 10:16:19

  Traffic      ------------inbound------------    ------------outbound-----------
               rcpts       msgs       kbytes        rcpts       msgs       kbytes
      Total   5100984    128853       1950.4       5313568    5313568    2787064.1
  Last Hour    295872      7373        111.6        295890     295890      97714.5
   Top/Hour    303999      7591        114.9        450637     450637    1176613.1
  Last Min.      6019       155          2.3          6017       6017       1994.1
   Top/Min.      8365       207          3.1        213222     213222    1098011.2


  Connections   active        top      maximum  Domain        cached       pending
     Inbound         0          1          150  Names              0             0
    Outbound         0        299          500

      Queues     rcpts     domains       kbytes  Spool         in use       recycled
        SMTP       639         239          9.7  Files            466              0
       other         0           0          0.0  Init.                       complete

      Status   running     Started  2015-08-31 16:22:01   Uptime   0 17:54:18
```

XML Format (`pmta --xml show status`):

```
<rsp><data><mta><product><name>PowerMTA</name><version>4.5</version><buildDat
e>2015-08-31
```

```
16:07:03</buildDate><revision>tags/v4_5c1@15693</revision><bits>64</bits></pr
oduct><lak><serial>538683</serial><instances>1</instances></lak><os><name>Lin
ux</name><version>3.10.0-229.1.2.el7.x86_64</version><build>CentOS     Linux
release                    7.1.1503                    (Core)
</build></os><cpu><type>x86_64</type><count>12</count></cpu><ram><real>327506
28</real></ram><fullHostName>example.port25.com</fullHostName><userString></u
serString><hostId>0</hostId></mta><status><timeNow>2015-09-01
10:17:07</timeNow><startupTime>2015-08-31
16:22:01</startupTime><shuttingDown>0</shuttingDown><traffic><total><out><rcp
>5317436</rcp><msg>5317436</msg><kb>2788394.0</kb></out><in><rcp>5104905</rcp
><msg>128951</msg><kb>1951.8</kb></in><bounceProcessed>0</bounceProcessed><fe
edbackLoopProcessed>0</feedbackLoopProcessed></total><lastHr><out><rcp>295376
</rcp><msg>295376</msg><kb>97626.3</kb></out><in><rcp>295358</rcp><msg>7366</
msg><kb>111.4</kb></in><bounceProcessed>0</bounceProcessed><feedbackLoopProce
ssed>0</feedbackLoopProcessed></lastHr><lastMin><out><rcp>4776</rcp><msg>4776
</msg><kb>1669.2</kb></out><in><rcp>4772</rcp><msg>124</msg><kb>1.9</kb></in>
<bounceProcessed>0</bounceProcessed><feedbackLoopProcessed>0</feedbackLoopPro
cessed></lastMin><topPerHr><out><rcp>450637</rcp><msg>450637</msg><kb>1176613
.1</kb></out><in><rcp>303999</rcp><msg>7591</msg><kb>114.9</kb></in><bouncePr
ocessed>0</bounceProcessed><feedbackLoopProcessed>0</feedbackLoopProcessed></
topPerHr><topPerMin><out><rcp>213222</rcp><msg>213222</msg><kb>1098011.2</kb>
</out><in><rcp>8365</rcp><msg>207</msg><kb>3.1</kb></in><bounceProcessed>0</b
ounceProcessed><feedbackLoopProcessed>0</feedbackLoopProcessed></topPerMin><p
erf><lastMin><in><count>0</count><mailFromTime>0</mailFromTime><dataTime>0</d
ataTime><totalTime>0</totalTime></in></lastMin></perf></traffic><conn><smtpIn
><cur>0</cur><max>150</max><top>1</top></smtpIn><smtpOut><cur>0</cur><max>500
</max><top>299</top></smtpOut></conn><resolver><namesCached>0</namesCached><q
ueriesPending>0</queriesPending><queriesActive>0</queriesActive></resolver><q
ueue><smtp><rcp>692</rcp><dom>241</dom><kb>10.4</kb></smtp><pipe><rcp>0</rcp>
<dom>0</dom><kb>0.0</kb></pipe><discard><rcp>0</rcp><dom>0</dom><kb>0.0</kb><
/discard><file><rcp>0</rcp><dom>0</dom><kb>0.0</kb></file><alias><rcp>0</rcp>
<dom>0</dom><kb>0.0</kb></alias><bounceProcessor><rcp>0</rcp><dom>0</dom><kb>
0.0</kb></bounceProcessor><feedbackLoopProcessor><rcp>0</rcp><dom>0</dom><kb>
0.0</kb></feedbackLoopProcessor></queue><spool><initPct>100</initPct><dirs>1<
/dirs><files><inUse>491</inUse><recycled>0</recycled><total>491</total></file
s><totalRcp>692</totalRcp><maxRcp>1000000</maxRcp></spool><status>running</st
atus></status></data><succ></succ></rsp>
```

DOM-Style (`pmta --dom show status`):

```
mta.product.name="PowerMTA"
mta.product.version="4.5"
mta.product.buildDate="2015-08-31 16:07:03"
mta.product.revision="tags/v4_5@15693"
mta.product.bits="64"
mta.lak.serial="538683"
mta.lak.instances="1"
mta.os.name="Linux"
mta.os.version="3.10.0-229.1.2.el7.x86_64"
mta.os.build="CentOS Linux release 7.1.1503 (Core) "
mta.cpu.type="x86_64"
mta.cpu.count="12"
mta.ram.real="32750628"
mta.fullHostName="example.port25.com"
mta.userString=""
mta.hostId="0"
status.timeNow="2015-09-01 10:17:50"
```

```
status.startupTime="2015-08-31 16:22:01"
status.shuttingDown="0"
status.traffic.total.out.rcp="5319661"
status.traffic.total.out.msg="5319661"
status.traffic.total.out.kb="2789142.9"
status.traffic.total.in.rcp="5107093"
status.traffic.total.in.msg="129008"
status.traffic.total.in.kb="1952.7"
status.traffic.total.bounceProcessed="0"
status.traffic.total.feedbackLoopProcessed="0"
status.traffic.lastHr.out.rcp="294091"
status.traffic.lastHr.out.msg="294091"
status.traffic.lastHr.out.kb="97231.2"
status.traffic.lastHr.in.rcp="294046"
status.traffic.lastHr.in.msg="7335"
status.traffic.lastHr.in.kb="111.0"
status.traffic.lastHr.bounceProcessed="0"
status.traffic.lastHr.feedbackLoopProcessed="0"
status.traffic.lastMin.out.rcp="3275"
status.traffic.lastMin.out.msg="3275"
status.traffic.lastMin.out.kb="1092.6"
status.traffic.lastMin.in.rcp="3242"
status.traffic.lastMin.in.msg="85"
status.traffic.lastMin.in.kb="1.3"
status.traffic.lastMin.bounceProcessed="0"
status.traffic.lastMin.feedbackLoopProcessed="0"
status.traffic.topPerHr.out.rcp="450637"
status.traffic.topPerHr.out.msg="450637"
status.traffic.topPerHr.out.kb="1176613.1"
status.traffic.topPerHr.in.rcp="303999"
status.traffic.topPerHr.in.msg="7591"
status.traffic.topPerHr.in.kb="114.9"
status.traffic.topPerHr.bounceProcessed="0"
status.traffic.topPerHr.feedbackLoopProcessed="0"
status.traffic.topPerMin.out.rcp="213222"
status.traffic.topPerMin.out.msg="213222"
status.traffic.topPerMin.out.kb="1098011.2"
status.traffic.topPerMin.in.rcp="8365"
status.traffic.topPerMin.in.msg="207"
status.traffic.topPerMin.in.kb="3.1"
status.traffic.topPerMin.bounceProcessed="0"
status.traffic.topPerMin.feedbackLoopProcessed="0"
status.traffic.perf.lastMin.in.count="0"
status.traffic.perf.lastMin.in.mailFromTime="0"
status.traffic.perf.lastMin.in.dataTime="0"
status.traffic.perf.lastMin.in.totalTime="0"
status.conn.smtpIn.cur="0"
status.conn.smtpIn.max="150"
status.conn.smtpIn.top="1"
status.conn.smtpOut.cur="0"
status.conn.smtpOut.max="500"
status.conn.smtpOut.top="299"
status.resolver.namesCached="0"
status.resolver.queriesPending="0"
```

166

```
status.resolver.queriesActive="0"
status.queue.smtp.rcp="655"
status.queue.smtp.dom="243"
status.queue.smtp.kb="9.9"
status.queue.pipe.rcp="0"
status.queue.pipe.dom="0"
status.queue.pipe.kb="0.0"
status.queue.discard.rcp="0"
status.queue.discard.dom="0"
status.queue.discard.kb="0.0"
status.queue.file.rcp="0"
status.queue.file.dom="0"
status.queue.file.kb="0.0"
status.queue.alias.rcp="0"
status.queue.alias.dom="0"
status.queue.alias.kb="0.0"
status.queue.bounceProcessor.rcp="0"
status.queue.bounceProcessor.dom="0"
status.queue.bounceProcessor.kb="0.0"
status.queue.feedbackLoopProcessor.rcp="0"
status.queue.feedbackLoopProcessor.dom="0"
status.queue.feedbackLoopProcessor.kb="0.0"
status.spool.initPct="100"
status.spool.dirs="1"
status.spool.files.inUse="476"
status.spool.files.recycled="0"
status.spool.files.total="476"
status.spool.totalRcp="655"
status.spool.maxRcp="1000000"
status.status="running"
```

6.3 Command Reference

6.3.1 `clear dnscache`

Syntax:

```
pmta clear dnscache domain
```

Description:

This command clears PowerMTA's internal DNS cache, removing all entries for the given domain. If the domain is * all cached DNS data is discarded. On Unix the asterisk must be escaped with a backslash.

Example:

```
$ pmta clear dns \*
DNS cache cleared.

$ pmta clear dns port25.com
Cleared port25.com from DNS cache.
```

6.3.2 `check mfrom`

Syntax:

```
pmta check mfrom [--tcp] [--dumppackets] mfrom ip
```

Description:

This command checks whether a message with the given "mfrom" (address in the SMTP `MAIL FROM` command, or, if empty, the domain passed in the `HELO` command) would pass the Sender-ID check if sent from the given IP address. It is useful to quickly verify SPF 2.0 records with the "mfrom" scope.

Options:

`--tcp`

> use TCP (rather than the default UDP) for querying the name server(s)

`--dumppackets`

> print out an hexadecimal dump of any DNS queries and responses exchanged with the name servers. It is ignored unless the output is in text format.

6.3.3 `reload`

Syntax:

```
pmta reload
```

Description:

PowerMTA reads its configuration during startup. Use this command to activate the new settings after making changes to the configuration file. The `reload` command is generally preferrable to stopping and re-starting PowerMTA because it is more efficient and does not cause an interruption in service. However, some of the configuration directives cannot be reloaded dynamically and require a restart. See the directive's attributes in [Section 3.2](#) to find out whether it is dynamically reloadable.

If PowerMTA detects an error in the configuration file, the `reload` command fails and the previous in-memory configuration is retained.

Example:

```
$ pmta reload
Configuration reloaded.
```

6.3.4 `reset counters`

Syntax:

```
pmta reset counters
```

Description:

PowerMTA performs various statistics based on traffic counters that can can be viewed from the web monitor and with the `show status` command. Normally these statistics include all traffic since PowerMTA was started up, but the `reset counters` command

allows you to reset these counters without stopping and re-starting PowerMTA. Before resetting, PowerMTA saves the counters to the accounting file.

Example:

```
$ pmta reset count
Traffic counters reset.
```

6.3.5 `reset status`

Syntax:

```
pmta reset status
```

Description:

This command is just an alias to `reset counters`.

6.3.6 `resolve`

Syntax:

```
pmta resolve [--tcp] [--connect] [--dumppackets] [--interactive] [--source=[host,]ip] domainname
```

Description:

The `resolve` command queries the DNS for mail routing information (MX and A records) for the specified domain, displays that information, and optionally connects to the first available mailer. It is intended to facilitate tracking down connectivity problems to destination domains. For example, if you notice that the queue is building up for a specific domain, you can run `resolve` against that domain to figure out what problems exist and what if anything can be done to fix the problem.

Defaults:

If no options are specified, DNS resolution is attempted via UDP and any relevant information is displayed, however no attempt is made to connect to a mailer.

Options:

`--tcp`

> this tells PowerMTA to use TCP (rather than the default UDP) for querying the name server(s)

`--connect`

> this option tells PowerMTA to attempt to connect to the first available mailer as well, after querying and displaying the specified domain's DNS server for mail routing information.

`--dumppackets`

> this option tells PowerMTA to print out an hexadecimal dump of any DNS queries and responses exchanged with the name servers. It is ignored unless the output is in textual format.

`--interactive`

> this option allows the user to try various SMTP commands, to help diagnose problems.

`--source`

> this option tells PowerMTA form which internal IP address or hostname to

169

connect.

Examples:

```
$ pmta resolve --connect hotmail.com
Querying 192.54.35.100 over UDP about MX(15) hotmail.com
Querying 192.54.35.100 over UDP about A(1) mc4.law5.hotmail.com
Querying 192.54.35.100 over UDP about A(1) mc5.law5.hotmail.com
Querying 192.54.35.100 over UDP about A(1) mc2.law5.hotmail.com

status = StatusOk (no error)
pref- host name-------------------- IP addresses/resolution status-----------
   10 mc4.law5.hotmail.com          216.33.151.136
   10 mc5.law5.hotmail.com          216.32.243.136
   10 mail.hotmail.com              216.33.151.135
   10 mc2.law5.hotmail.com          216.32.243.135

Connecting to mc4.law5.hotmail.com (216.33.151.136)... ok
>>> 220-HotMail (NO UCE) ESMTP server ready at Tue Jun 27 01:48:47 2003
>>> 220 ESMTP spoken here
<<< ehlo test
>>> 250-hotmail.com Hello
>>> 250 SIZE 1048576
<<< quit
>>> 221 Service closing transmission channel

$ pmta --dom resolve yahoo.com
Querying 192.54.35.100 over UDP about MX(15) yahoo.com
Querying 129.26.8.82 over UDP about MX(15) yahoo.com
Querying 192.54.35.100 over UDP about A(1) mx1.mail.yahoo.com
Querying 192.54.35.100 over UDP about A(1) mx2.mail.yahoo.com
status="StatusOk (no error)"
host[0].pref="0"
host[0].name="mx1.mail.yahoo.com"
host[0].status="StatusOk (no error)"
host[0].ip[0]="128.11.23.224"
host[0].ip[1]="128.11.68.143"
host[0].ip[2]="128.11.68.214"
host[0].ip[3]="128.11.68.223"
host[0].ip[4]="128.11.23.236"
host[0].ip[5]="128.11.23.247"
host[0].ip[6]="128.11.23.230"
host[0].ip[7]="128.11.23.235"
host[1].pref="1"
host[1].name="mx2.mail.yahoo.com"
host[1].status="StatusOk (no error)"
host[1].ip[0]="128.11.68.148"
host[1].ip[1]="128.11.23.227"
host[1].ip[2]="128.11.23.236"
host[1].ip[3]="128.11.68.145"
host[1].ip[4]="128.11.23.229"
host[1].ip[5]="128.11.68.95"
host[1].ip[6]="128.11.23.230"
host[1].ip[7]="128.11.68.213"
```

6.3.7 `rotate acct`

Syntax:

```
pmta rotate acct [file]
```

Description:
This command instructs PowerMTA to immediately rotate the accounting file. If included, only the specified file will be rotated. Immediate rotation does not affect the automatic rotation at midnight, which is still performed.

Example:

```
C:\>C:\pmta\bin\pmta rotate acct
Accounting file rotated.
```

6.3.8 `rotate log`

Syntax:

```
pmta rotate log
```

Description:
This command instructs PowerMTA to immediately rotate the logging file. Immediate rotation does not affect the automatic rotation at midnight, which is still performed.

Example:

```
C:\>C:\pmta\bin\pmta rotate log
Logging file rotated.
```

6.3.9 `schedule`

Syntax:

```
pmta schedule [--retry-recipients] domain/vmta
```

Description:
Instructs the mailer to immediately open a connection to the specified domain and to attempt delivery of messages queued to it. This command would be used, for example, if one figured out and fixed connectivity problems to a site and then wanted to bypass the normal retry interval from the configuration file. If there are no entries in the queue for the domain specified, the PowerMTA's response says so and it does not connect to the domain. A wildcard can also be specified in *domain* or *vmta*, allowing multiple domains/vmtas to be scheduled at once.

Options:

`--retry-recipients`

Overrides the 'retry-after' period for such recipients, and immediately makes them available for delivery attempts

`--vmta=name`

specifies the VirtualMTA from which to open the connection. If not specified, all VirtualMTAs for which messages are queued are scheduled. Only specific

171

VirtualMTAs are accepted (no VirtualMTA pools).

Example:
```
$ pmta schedule "port25.com"
0 of 0 matching, 0 total domains scheduled.
```

6.3.10 `set queue`

Syntax:
```
pmta set queue --mode={normal|backoff} domain/vmta
```

Description:

Sets the given queue's mode of operation to either `normal` or `backoff`. While in backoff mode, various different settings may apply. See the `backoff-...` directives in Section 3.2.8 for more information.

Wildcards can be used to modify various queues at the same time, but when used, they only apply to queues that contain messages or that are already in backoff mode.

6.3.11 `show domains`

Syntax:
```
show domains [--vmta=name] [--connected={yes|no}] [--maxitems=n]
             [--errors] [--sort ={name|rcpt|size}] [name]
```

Description:

This command allows you to view current information on the specified destination domain(s).

Options:

--vmta=*name*

  displays only domains information for the given VirtualMTA. The default is to display (accumulated) information for all VirtualMTAs, including messages for which no VirtualMTA was selected.

--connected

  displays domains to which PowerMTA has least one open outgoing connection. The default is to display both connected and unconnected domains.

--maxitems=*n*

  specifies the maximum number of domains to be displayed in the output. By default, 20 domains are displayed.

--errors

  displays any errors recorded for the domain.

--sort=*key*

  sort ouput by *key* where *key* may be *name* to order the domain list alphabetically by name, *rcpt* to order the list by number of recipients or *size* to order the list by volume (kbytes).

name

  domain name or wildcard specifying the domains about which information is to

172

be displayed. If omitted, all domains are displayed (subject to the `maxitems` limit).

Examples:

```
$ pmta show dom
---------------------------domain --#rcpt ---kbytes conn last error---------
                 test1.port25.com      7      30.6   0
                test10.port25.com      8      30.6   0
                test11.port25.com      3      30.6   0
                test12.port25.com     13      30.6   0
                test13.port25.com      3      30.6   0
                test14.port25.com      3      30.6   0
                test15.port25.com      3      30.6   0
                test16.port25.com     52      30.6   0
                test17.port25.com      3      30.6   0
                test18.port25.com      3      30.6   0 ETIMEDOUT connect...
                test19.port25.com      3      30.6   0
                 test2.port25.com      3      30.6   0
                test20.port25.com      3      30.6   0
                test21.port25.com      3      30.6   0
                test22.port25.com      3      30.6   0
                test23.port25.com      3      30.6   0
                test24.port25.com      3      30.6   0
                test25.port25.com      3      30.6   0
                test26.port25.com      3      30.6   0
                test27.port25.com      3      30.6   0

20 of 30 domains shown.

$ pmta show dom --conn
0 of 0 matching, 30 total domains.
```

6.3.12 `show queues`

Syntax:

```
show queues [--connected={yes|no}] [--paused={yes|no}]
            [--mode={normal|backoff}] [--maxitems=n] [--errors]
            [--sort= {name|rcpt|size}] [--no-rcpt-events] [queue] [name]
```

Description:

This command allows you to view current information on the specified queues. The output includes the last 50 delivery/bounce/deferral events

Options:

--connected

> displays queues for which PowerMTA has least one open outgoing connection. The default is to display both connected and unconnected queues.

--mode

> displays only queues in the specified mode. The default is to display queues in all modes.

--maxitems=n

> specifies the maximum number of queues to be displayed in the output. By default, 20 domains are displayed.

--errors

173

displays any errors recorded for the queue.

--sort=*key*

sort ouput by *key* where *key* may be `name` to order the queue list alphabetically by name, `rcpt` to order the list by number of recipients or `size` to order the list by volume (kbytes).

--no-rcpt-events

hides the delivery events, that by default are shown.

--paused={yes|no}

allows selecting only paused or running queues for display. By default, both are shown.

queue

name of the queue(s) to display, in the format *domain[/vmta]*, where *domain* is the destination domain name and *vmta* is the VirtualMTA. If omitted, both *domain* and *vmta* default to * (i.e., `*/*`), which matches all queues.

174

Examples:

```
$ pmta show queue
queue          #rcpt kbytes conn st    retry last error
-------------- ----- ------ ---- -- -------- ----------
*/*
example.com/vmta1     2    0.2    0    09:35:42

  Recipient Events (deliveries:4, bounces:1, deferrals:1)
  -----------------------------------------------------
  delivered | 2014-11-24 09:35:09 | 10.25.25.235 | kf@example.com | jobid10145783566176 | [192.168.10.48] (192.168.10.48) |
smtp;250 DATA ok
  delivered | 2014-11-24 09:31:07 | 10.25.25.235 | ft@example.com | jobid10145783566176 | [192.168.10.48] (192.168.10.48) |
smtp;250 DATA ok
  deferred  | 2014-11-24 09:29:06 | 10.25.25.235 | np@example.com | jobid8363372138340 | [192.168.10.48] (192.168.10.48) |
smtp;450 4.0.0 (undefined status);smtp;421 Refused. Your reverse DNS entry does not resolve.
  delivered | 2014-11-24 09:26:06 | 10.25.25.235 | ce@example.com | jobid8363372138340 | [192.168.10.48] (192.168.10.48) |
smtp;250 DATA ok
  delivered | 2014-11-24 09:26:06 | 10.25.25.235 | rd@example.com | jobid10145783566176 | [192.168.10.48] (192.168.10.48) |
smtp;250 DATA ok
  bounced   | 2014-11-24 09:36:09 | 10.25.25.235 | vc@example.com | jobid8363372138340 | [192.168.10.48] (192.168.10.48) |
smtp;550 5.0.0 (undefined status);smtp;550 No Such User Here3 of 3 matching queues.
```

Note that in the example above, there are two queues for `port25.com`: one for the default VirtualMTA, and another for the `mta1` VirtualMTA.

```
$ pmta show que --mode=backoff
----------------------------queue --#rcpt ---kbytes conn ---mode last
error--
                    port25.com/mta1     2       0.3    0 backoff

1 of 1 matching queue.
```

6.3.13 `show jobs`

Syntax:

```
show jobs [--maxitems=n] domain[/vmta]
```

Description:

This command allows you to view the jobs currently in PowerMTA's queue. Jobs are groups of messages. You can tell PowerMTA that various messages are in the same job by specifying the job ID

- in the `x-job` header (if feeding through SMTP, you should first enable `x-job` processing -- see Section 3.2.4 for details)
- by specifying an envelope ID in the format `pmta-XXXX-...` where `XXXX` is the job ID
- by calling `setJobId` in the submission API

175

You can also get the jobs information in XML format from the web monitor directly. See Section 5.2 for details.

Options:

--maxitems=*n*

specifies the maximum number of queues to be displayed in the output. By default all jobs are displayed.

domain[/vmta]

name of the queue(s) to display, in the format *domain[/vmta]*, where *domain* is the destination domain name and *vmta* is the VirtualMTA. If omitted, both *domain* and *vmta* default to * (i.e., */*), which matches all queues.

Example:

```
$ pmta show jobs
----------------------------job-id ----#rcpt ---kbytes
                  weeklyNews22100        431     1236.9

Total of 1 job(s).
```

6.3.14 `show settings`

Syntax:

```
show settings domain[/vmta]
```

Description:

This command allows you to view the (per-queue) settings for the specified domain and VirtualMTA.

Examples:

```
$ pmta -dom show settings port25.com/vmta1
Virtual MTA Settings
-------------------
max-smtp-msg-rate unlimited
max-smtp-out unlimited

Queue Settings
--------------
allow-cancel-during-transfer="yes"
auth-password=""
auth-username=""
backoff-max-msg-per-hour="unlimited"
backoff-notify="no"
backoff-reroute-to-virtual-mta=""
backoff-retry-after="1h"
bounce-after="1d"
bounce-upon-5xx-greeting="yes"
bounce-upon-pix-transfer-failure="no"
bounce-upon-transfer-failure="no"
command=""
connect-timeout="2m"
data-send-timeout="3m"
```

176

```
queue-priority="50"
file-destination=""
file-format="newfile-plain"
ignore-8bitmime="no"
log-commands="no"
log-connections="no"
log-data="no"
log-resolution="no"
log-transfer-failures="no"
max-events-recorded="10"
max-msg-per-connection="0"
max-msg-per-hour="unlimited"
max-rcpt-per-message="1000"
max-smtp-out="20"
pix-bug-workaround="yes"
retry-after="10m"
route=""
smtp-pattern-list=""
type="file"
```

6.3.15 `show status`

Syntax:

```
pmta show status
```

Description:

Displays a snapshot of what is currently and has previously taken place within the mailer since last restart. Intended for a allowing a "quick look" into the general status. The output includes totals, top throughput and queue information among other things. The output is equivalent to the data available via the web-based monitoring console. You can also get the same information in XML format from the web monitor directly. See Section 5.2 for details.

Example:

```
$ pmta show status
        PowerMTA v2.1a5 status on hazmat.port25.com on 2003-05-25 21:26:48

  Traffic      ------------inbound------------  ------------outbound-----------
              rcpts       msgs      kbytes        rcpts       msgs      kbytes
      Total        0          0        0.0            0          0        0.0
  Last Hour        0          0        0.0            0          0        0.0
   Top/Hour        0          0        0.0            0          0        0.0
  Last Min.        0          0        0.0            0          0        0.0
   Top/Min.        0          0        0.0            0          0        0.0

Connections     active        top     maximum  Domain         cached       pending
   Inbound          0          0          30   Names              0             0
  Outbound          0          0         800

     Queues      rcpts    domains      kbytes  Spool          in use      recycled
       SMTP          0          0        0.0   Files              0             0
      other          0          0        0.0   Init.                      complete

     Status    running    Started  2003-05-25 21:24:16   Uptime    0   0:02:32
```

177

6.3.16 `show topdomains`

Syntax:

```
pmta show topdomains [--vmta=name] [--connected={yes|no}]
                     [--maxitems=n] [--errors]
```

Description:

For displaying the top domains in the queue sorted by number of recipients. Since this command is equivalent to `show domains --sort=rcpt`, please see Section 6.3.12 for more information on the various options.

6.3.17 `show topqueues`

Syntax:

```
show topqueues [--connected={yes|no}] [--mode={normal|backoff}]
               [--maxitems=n] [--errors] [--paused={yes|no}]
               [--no-rcpt-events] [domain[/vmta]
```

Description:

For displaying the top queues sorted by number of recipients. Since this command is equivalent to `show queues --sort=rcpt`, please see Section 6.3.13 for more information on the various options.

6.3.18 `show version`

Syntax:

```
pmta show version
```

Description:

Displays PowerMTA's version and build date.

Example:

```
$ pmta show ver
PowerMTA v2.0r1, built on Jun 27 2003 18:36:16.
```

6.3.19 `show vmtas`

Syntax:

```
show vmtas [--maxitems=n]
```

Description:

This command displays a summary of the VirtualMTA(s) in use, as well as the number of recipients and kilobytes in messages which select it.

Options:

`--maxitems=n`

> specifies the maximum number of queues to be displayed in the output. By default all vmtas are displayed.

Example:

178

```
$ pmta show vmtas
----------------------------name ----#rcpt ---kbytes
                            mta1        4       0.3
                            mta2        2       0.1

Total of 2 virtual MTA(s).
```

6.3.20 `trace`

Syntax:
```
pmta trace [--log-data] [--log-resolution] [--to-log]
           [--retry-recipients] [--source-ip=ip] domain/vmta
```

Description:

Instructs the mailer to open a connection to the specified domain and to attempt delivery of any messages queued to it. This command is similar to the `schedule` command, with the difference that it also automatically enables logging for the domain for the duration of one connection. If the –to-log switch is used, it is a shortcut to enabling logging in the configuration file, reloading the configuration, scheduling a call and undoing these changes when debugging is finished. By default, only the SMTP commands and responses are logged. The result of this command is written to the logging file.

Options:

`--log-data`

> this enables logging of data transfers as well, like the `log-data` option in the configuration file.

`--log-resolution`

> this enables logging of the DNS resolution as well, like the `log-resolution` option in the configuration file.

`--to-log`

> this enables logging trace info to the log file

`--retry-recipients`

> Use of the retry-recipients flag will cause all the recipients of the queue that are in Await Retry bucket to become available for retry immediately.

`--source-ip`

> Specifies the source IP used for connection attempts from a virtual mta with multiple smtp-source-hosts configured in it

`domain[/vmta]`

> name of the queue(s) to display, in the format `domain[/vmta]`, where *domain* is the destination domain name and *vmta* is the VirtualMTA. If omitted, both *domain* and *vmta* default to `*` (i.e., `*/*`), which matches all queues.

Example:
```
$ pmta trace yahoo.com
Traced connection scheduled (see logging file).
```

Logging file output:

179

```
2003-06-27 14:03:10 (402)starting yahoo.com
2003-06-27 14:03:10 (402)connecting to mx1.mail.yahoo.com (128.11.68.141)
2003-06-27 14:03:11 (402)>>> 220 YSmtp mta26.mail.yahoo.com ESMTP service ready
2003-06-27 14:03:11 (402)<<< EHLO hazmat.port25.com
2003-06-27 14:03:11 (402)>>> 250-mta26.mail.yahoo.com
2003-06-27 14:03:11 (402)>>> 250-8BITMIME
2003-06-27 14:03:11 (402)>>> 250-SIZE 3145728
2003-06-27 14:03:11 (402)>>> 250 PIPELINING
2003-06-27 14:03:11 (402)<<< QUIT
2003-06-27 14:03:11 (402)>>> 221 mta26.mail.yahoo.com
2003-06-27 14:03:11 (402)done yahoo.com in=155 out=30
```

6.3.21 `delete`

Syntax:

```
pmta delete [--dsn] [--queue=domain[/vmta] [--orig=addr] [--rcpt=addr]
             [--jobId=id] [--envId=id] [--accounting]
             [--older-than=time-interval]
```

Description:

Deletes recipients from the queue. Although "--queue"'s default is */*, the command requires either --queue or --jobId to be entered explicitly, so that an accidental "pmta delete" command doesn't delete the entire queue. Messages are marked in the accounting file as deleted by administrator.

Options:

`--dsn`

> specifies that a DSN report should be sent. Using this option may slow the deletion process. No entry is written to the accounting file unless the --accounting flag is used as well.

`--queue`

> specifies which queue(s) to delete from, and defaults to all queues.

`--orig`

> specifies that only recipients with the MAIL FROM address be deleted

`--rcpt`

> specifies that only recipients with the RCPT TO address are to be deleted.

`--jobId`

> specifies that only recipients with the given Job ID are to be deleted

`--envId`

> specifies that only recipients with the given Envelope ID are to be deleted.

`--accounting`

> specifies that deletion information should be written to the accounting file. Using this option may slow the deletion process.

`--older-than`

> deletes all the recipients in the given queue that are older than the given from the current time. The syntax for the time interval is same as the syntax used to specify bounce-after in the configuration file.

Example:

180

```
$ pmta delete --queue=yahoo.com/*
```

6.3.22 `pause`

Syntax:
```
pmta pause queue domain[/vmta]
pmta pause job jobid
```
Description:
Pause delivery of messages in the queue, and holds them from being delivered. May be run prior to injecting mail into PowerMTA so that mail may be held for later delivery.
Example:
```
$ pmta pause queue */customer1
$ pmta pause job campaign1234
```

6.3.23 `resume`

Syntax:
```
pmta resume queue domain[/vmta]
pmta resume job jobid
```
Description:
Resumes delivery of messages in the queue.
Example:
```
$ pmta resume queue */customer1
$ pmta resume job campaign1234
```

6.3.24 `list`

Syntax:
```
pmta list [--queue=domain[/vmta]] [--orig=addr] [--rcpt=addr]
[--jobId=id] [--envId=id] [--maxitems=n] [--pause] [--priority]
```
Description:
Allows listing recipients from the command line.
Options:
--queue
> specifies which queue(s) to list from, and defaults to all queues.

--orig
> specifies that only recipients with the MAIL FROM address be listed

--rcpt
> specifies that only recipients with the RCPT TO address are to be listed.

--jobId
> specifies that only recipients with the given Job ID are to be listed

--envId
> specifies that only recipients with the given Envelope ID are to be listed.

--maxitems=*n*
> specifies the maximum number of queues to be displayed in the output. By default 100 recipients are displayed.

--pause
> if recipient is being actively delivered, pause the delivery to enable showing the recipient in the output

--priority
> display the priority of each recipient in the output of the command

--schedule
> display the per-recipient schedule in the output of the command

Example:
```
$ pmta list --queue=yahoo.com/vmta1
```

6.3.25 `license`

Syntax:

```
pmta show license
```

Description:

Allows listing license information from the command line.

Example:

```
$ pmta show license
  product: PowerMTA
  version: 3.5
 platform: linux-intel
    units: 0
  options:
 licensee: Acme Corp.
   serial: 123456789
  comment:
   issued: 2008-03-26
  expires: 2008-04-36
copyright: Port25 Solutions, Inc.  All Rights Reserved
```

6.3.26 `deregister`

Syntax:

```
pmta deregister [--user=name] [--local-only] [--retain-unloaded]
```

Description:

Allows de-registering a node for PowerMTA Management Console. This command will delete all data that has not yet been loaded into PowerMTA Management Console. This does not effect the user configured accounting files.

Options:

`--user`

> specifies the PowerMTA Management Console user account to use when deregistering

--local-only

> only dergisters locally, registration will still exist in PowerMTA Management Console.

--retain-unloaded

> prevents deleting accounting files that have not yet been loaded from PowerMTA into PMC. Useful when needing to deregidster and then reregister without losing any data.

Example:
```
$ pmta deregister --retain-unloaded
Deregistration successful.
You may still need to remove this PowerMTA node manually from
PowerMTA Management Console.
```


6.3.27 `register`

Syntax:
```
pmta register [--user=name] [--label=name] [--webmon-ip=ip] [--
webmon-port=number]    [--pmtamc-port=number]    [--reuse-label]
pmtamc-hostname
```

Description:
Allows registering a node for PowerMTA Management Console

Options:
--user
> PowerMTA Management Console user to be used for registration

--label
> Name of PowerMTA instance that will be shown in PowerMTA Management Console

--webmon-ip
> IP address for PowerMTA Management Console to connect to PowerMTA instance

--webmon-port
> Port for PowerMTA Management Console to connect to PowerMTA instance

--pmtamc-port
> PowerMTA Management Console port to be used for registration

--reuse-label
> If label was previously used, but deregistered, this allows label to be used again

--pmtamc-hostname
> PowerMTA Management Console server to be used for registration

Example:

```
$ pmta register -label=pmta1server central.yourdomain.com
PowerMTA Management Console "admin" password:
Registration successful.
PowerMTA Management Console certificate fingerprint:
    0B:1D:EC:9C:8F:20:4E:06:55:2C:A8:9B:A1:A6:05:C7:DD:07:93:01

For best security, please verify that the above value is the same as
the  one  displayed  on  PowerMTA  Management  Console's  PowerMTA
Management page.
```

6.3.28 `show registration`

Syntax:

```
pmta show registration
```

Description:

Shows current PowerMTA Management Console registration informaiton.

Example:

```
$ pmta show registration
Currently registered with PowerMTA Management Console
cental.yourdomain.com:8181 as "pmta1server"
PowerMTA Management Console certificate fingerprint:
    0B:1D:EC:9C:8F:20:4E:06:55:2C:A8:9B:A1:A6:05:C7:DD:07:93:01
```

6.3.29 `disable source`

Syntax:

```
pmta disable source [--reenable-after=interval] ip domain[/vmta]
```

Description:

Disables a given source IP from being used.

Example:

```
$ pmta disable source --reenable-after=1h 1.2.3.4 yahoo.com/vmta1
```

6.3.30 `enable source`

Syntax:

```
pmta enable source ip domain[/vmta]
```

Description:

Enables a given source IP so it can be used.

Example:

```
$ pmta enable source 1.2.3.4 yahoo.com/vmta1
```

6.3.31 `show disabled sources`

Syntax:

```
pmta show disabled sources [domain[/vmta]]
```

Description:

Shows currently disabled source IPs.

Example:

```
$ pmta show disabled sources yahoo.com/vmta1
```

6.3.32 `show precache`

Syntax:

```
pmta show precache
```

Description:

Shows currently precached domains

Example:

```
$ pmta show precache
```

6.3.33 `set priority`

Syntax:

```
pmta set priority [--queue=domain[/vmta]] [--orig=addr]
               [--rcpt=addr] [--jobId=id] [--envId=id] number
```

Description:

Sets priority for a queue, job, or individual message

Example:

```
$ pmta set priority --rcpt=ceo@example.com 100
```

186

7. Application Programming Interfaces

PowerMTA includes various programming interfaces (APIs) intended to facilitate the creation of e-mail enabled applications and to optimize the way in which messages are submitted to the mailer. The following APIs are available:

- submission APIs, for submitting e-mail from Perl, Java, .NET, C++ and C programs;
- a "pickup" directory, for submitting preformatted e-mail by copying it into a directory;
- "pipe" delivery, for delivering e-mail to a local program;

The Cold Fusion tag from previous PowerMTA versions has been removed, see Section 7.6 below for details.

Use of the APIs may require the use of the always-allow-api-submission directive. See Section 3.2.13 for more information.

On Windows Sever, any applicable API materials can be found in the `api` subdirectory within the folder into which PowerMTA was installed. On Linux, these can be found on `/opt/pmta/api`.

7.1 Submission APIs

The submission APIs allow you to programatically submit messages for delivery by PowerMTA. Since most users already have preformatted messages to send, the APIs mostly concern themselves with the transport of such messages. However, they do include a `Date:` header generator and a base64 encoder to facilitate the jobs of those creating messages from scratch.

The submission APIs are object-oriented and comprise three objects:

- a `Connection` which represents a connection to a PowerMTA server;
- a `Message` which represents the message being submitted;
- a `Recipient` which represents the message's recipients.

The general procedure for submitting a message is pretty straightforward:

- connect to a PowerMTA server by creating a `Connection`;
- create a `Message`;
- create one or more `Recipient`s and add them to the `Message`;

187

- add headers and body to the `Message` by invoking the `addData` method one or more times;
- submit the message by invoking the `Connection`'s `submit` method, passing the `Message` to it.

Other methods are available to set various delivery options, select automatic data encoding, etc. See the language-specific sections below for more information.

7.1.1 Requirements

The submission APIs can be run remotely or locally, as they use a modified version of SMTP.

Applications using the submission APIs must also have the proper authorization. On Windows Server, programs must run under either the `SYSTEM` account or under an account belonging to the `Administrators` group.

On Unix, programs must either run as `root` or from an username who belongs to the `pmta` group. The `pmta` group is created automatically during PowerMTA's installation.

The following are required when using the PERL, .NET, or Java APIs:

- PERL 5.12 or newer when using the PERL API

- .NET 2.0 or better when using the .NET API

- Java 6 or better (for JavaMail: JavaMail and Java Activation Framework from Sun's download site as detailed in the JavaDoc from the archive) when using the Java API

- PowerMTA needs to be configured with "always-allow-relaying yes" and "allow-mailmerge yes" for the appropriate <source>. "always-allow-api-submission yes" can be used as well, as this will set "allow-mailmerge yes". If you want to use authentication you need to define users and passwords and allow auth for the source.

- It is recommend to add no more than 5000 recipients per batch submission. In the event that the batch fails, this will affect fewer recipients. Also, these helps keeps memory usage to a minimum.

7.1.2 Perl Submission API

Documentation for the perl APIs is provided in perldoc format. The APIs are available from the download site.

```
perldoc Port25::Submitter
perldoc Port25::Submitter::Connection
perldoc Port25::Submitter::Message
perldoc Port25::Submitter::Recipient
```

7.1.3 Java Submission API

Documentation for the Java APIs is provided in JavaDoc format.

On Windows, it can be found in the `pmtajavadoc.zip` file in the `api\java` subdirectory of where PowerMTA was installed.

On Linux, it can be found in the `/opt/pmta/api/java/doc/javadoc` directory as /opt/pmta/api/java/doc/javadoc/index.html.

Currently only Sun's Java SDK is supported. If you need to run it in a different environment, please contact Port25 Support at `support@port25.com`.

7.1.4 C++ Submission API

In C++, the three objects described in [Section 7.1](#) are available in the include files `submitter/Connection.hxx`, `submitter/Message.hxx` and `submitter/Recipient.hxx`.

On Windows, you can find these files in the `api\include` subdirectory of where PowerMTA was installed. On Linux, that is `/opt/pmta/api/include`.

All the methods in the submission API throw an exception in case of an error. Any errors render the object unusable, after which you should discard it.

7.1.4.1 Method Reference

Connection::Connection

Creates a new connection.

```
Connection(const char* server, int port, const char* name
= "", const char* password = "")
```

189

Arguments:

```
server
      server to which to connect, either as an numeric IP address or
      a host name.
port
      port number to connect to; use 25 for the default SMTP port.
name
      user name for authentication with the remote server. Leave name
      and password empty if you don't want to use authentification.
password
      password to use for authentication.   Authentication is done
      with CRAM-MD5, so no password is ever sent to the remote
      server.
```

Description:

Creates a new connection.

Connection::submit

Submits a message for delivery.

```
void Connection::submit(const Message& message)
```

Arguments:

```
message
        message to submit.
```

Description:

Submits a message for delivery.

Message::Message

Creates a new message.

```
Message::Message(const char* originator)
```

Arguments:

```
originator
        the new message's originator.
```

Description:

Creates a new message.

190

Message::setVerp

Sets whether this message should use the VERP extension.

```
void Message::setVerp(bool isVerp)
```

Description:

Sets whether this message should use the VERP extension.

Message::setReturnType

Sets the type of message return requested in delivery reports.

```
void Message::setReturnType(PmtaMsgRETURN type)
```

Arguments:

`type`

> This should be one of:

PmtaMsgRETURN_HEADERS	(default) return only this message's headers
PmtaMsgRETURN_FULL	return the full message headers and body

Description:

Sets the type of message return requested in delivery reports.

Message::setEnvelopeId

Sets this message's envelope ID.

```
void Message::setEnvelopeId(const char* envelopeId)
```

Arguments:

`envelopeId`

> envelope ID for this message.

Description:

Sets this message's envelope ID.

Message::setVirtualMta

Selects the VirtualMTA to use for this message.

191

```
void Message::setVirtualMta(const char* virtualMta)
```

Arguments:

```
virtualMta
```
  name of the VirtualMTA to use.

Description:
Selects the VirtualMTA to use for this message.

Message::setJobId

Sets the job id for this message.
```
void Message::setJobId(const char* jobId)
```

Arguments:

```
jobId
```
  id of the job, printable, non-white space characters only.

Description:
Sets the job id for this message. This tags the message as belonging to the specified job.

Message::addRecipient

Adds a recipient to this message.
```
void Message::addRecipient(const Recipient& r)
```

Arguments:

```
recipient
```
  Recipient to add

Description:
Adds a recipient to this message. Do not make any changes to a recipient after it has been added to the message.

Message::setEncoding

Sets the kind of encoding to perform on data.

192

```
void Message::setEncoding(PmtaMsgENCODING encoding)
```

Arguments:

```
encoding
```
> Data encoding desired. This should be either `PmtaMsgENCODING_7BIT`, `PmtaMsgENCODING_8BIT` or `PmtaMsgENCODING_BASE64`.

Description:
Sets the kind of encoding to perform on data. The encoding applies to all thereafter added data. The actual processing performed depends on the encoding selected: `PmtaMsgENCODING_7BIT` and `PmtaMsgENCODING_8BIT` both select the *identity* encoding, i.e., no transformation is performed on the data, except that any lines terminated with LF alone are converted to CRLF. In the case of 7-bit encoding, it is your responsibility to ensure that all bytes have their high bit off. `PmtaMsgENCODING_BASE64` specifies that any data added shall be encoded on the fly using base-64 encoding. No transformation is performed on the data prior to encoding, making it possible to transmit binary content. Note, however, that to effectively transmit binary data, you must also provide the appropriate MIME headers. In fact, the selected encoding should, in general, agree with the `Content-Transfer-Encoding` header you specify. The default (if this method is never invoked) is `PmtaMsgENCODING_7BIT`.

Message::beginPart

Starts the next mailmerge part which will have the given number.
```
void Message::beginPart(int partNum)
```

Arguments:

```
partNum
```
> positive integer for the part number. No part number may be specified more than once.

Description:
Starts the next mailmerge part which will have the given number. Note that part 1 is started automatically, so the first settable number is 2.

Message::addData

Appends data to this message.

```
void Message::addData(const char* data, int length)
```

Arguments:

```
data
```
  data to append.
```
length
```
  length of the data block.

Description:
Appends data to this message. Use this method to give this message both its headers and a body. How the data you add is handled depends on the kind of encoding selected (see `setEncoding`). All the data is entered as-is, with no mailmerge variable substitution being performed. To add mailmerge data, use `addMergeData`.

Message::addMergeData

Appends merge data to this message.
```
void Message::addMergeData(const char* data, int length)
```

Arguments:

```
data
```
  data to append.
```
length
```
  length of the data block.

Description:
Appends merge data to this message. Use this method to give this message both its headers and a body. How the data you add is handled depends on the kind of encoding selected (see `setEncoding`). Currently only 7-bit and 8-bit encodings are supported. If any mailmerge variables are included in the data, they will be substituted by their respective values.

Message::addData

Appends null-terminated data to this message.
```
void Message::addData(const char* data)
```

Arguments:

```
data
```

194

null-terminated data to append.

Description:
Appends null-terminated data to this message. This is a convenience method to facilitate adding C strings directly.

Message::addMergeData

Appends null-terminated merge data to this message.
```
void Message::addMergeData(const char* data)
```

Arguments:
```
data
```
null-terminated merge data to append.

Description:
Appends null-terminated merge data to this message. This is a convenience method to facilitate adding C strings directly.

Message::addDateHeader

Adds (appends) a `Date:` header to this message.
```
void Message::addDateHeader()
```

Description:
Adds (appends) a `Date:` header to this message. This method appends a date header line to the message. It is provided as a convenience function only.

Recipient::Recipient

Creates a new recipient with the given email address.
```
Recipient::Recipient(const char* address)
```

Arguments:
```
address
```
e-mail address for the recipient.

Description:

Creates a new recipient with the given email address.

Recipient::setNotify

Sets the kind of notification (report) desired for this recipient.

```
void Recipient::setNotify(int notifyWhen)
```

Arguments:

notifyWhen

This should be either `PmtaRcptNOTIFY_NEVER` to indicate that no notification is desired, or one or more of the following flags (joined by a bitwise OR):

PmtaRcptNOTIFY_SUCCESS	notify in case the delivery is successful
PmtaRcptNOTIFY_FAILURE	notify in case of a delivery failure
PmtaRcptNOTIFY_DELAY	notify in case of a delay in the delivery

Description:

Sets the kind of notification (report) desired for this recipient. By default, notification is only requested upon delivery failure. Note that while PowerMTA does not support notification for delivery delays, the corresponding `DSN` flag is passed on to the receiving mailer, which may support it.

Recipient::defineVariable

Defines a new mailmerge variable for this recipient.

```
void Recipient::defineVariable(char* name, char* value)
```

Arguments:

name
        the variable's name.
value
        the variable's value.

Description:

Defines a new mailmerge variable for this recipient. PowerMTA will substitute the variable for its value during delivery of a mailmerge message.

Carriage returns and line feeds may be added to a variable with \r or \n respectively.

```
Recipient rcpt("me@here.com");
rcpt.defineVariable("cr", "A\rB");      // one linebreak
```

196

```
rcpt.defineVariable("lf", "A\nB");       // one linebreak
rcpt.defineVariable("crLf", "A\r\nB");  // one linebreak

rcpt.defineVariable("literalCrLf", "A\\r\\nB");  // no linebreaks
```

7.1.5 C Submission API

In C, the three objects described in [Section 7.1](#) are available as sets of functions which share a common prefix. `Connection`'s functions all begin with `PmtaConn` and are defined in the file `submitter/PmtaConn.h`; `Message`'s functions all begin with `PmtaMsg` and are defined in the file `submitter/PmtaMsg.h`; and `Recipient`'s functions all begin with `PmtaRcpt` and are defined in the file `submitter/PmtaRcpt.h`.

On Windows, you can find these files in the `api\include` subdirectory of where PowerMTA was installed. On Linux, that is `/opt/pmta/api/include`.

In order to differentiate between "out of memory" conditions and other errors, object creation is split in two calls: allocation and initialization. To create an object, you fist call its ...`Alloc` function. If memory cannot be allocated, a null pointer is returned; otherwise, a pointer to the object is returned, which you then pass to the object's initialization function: `PmtaConnConnect`, `PmtaMsgInit` and `PmtaRcptInit`, for a `Connection`, `Message` or `Recipient` object, respectively.

When done with an object, you should call the object's ...`Free` method to discard its resources.

Most of the functions in the submission API return a `BOOL` (boolean) value. You should check every return value to make sure that no errors occurred. If an error occurs, you can find out what went wrong by calling either `PmtaConnGetLastError`, `PmtaMsgGetLastError` or `PmtaRcptGetLastError` (depending on which object the error occurred). Any errors render the object unusable, after which you should discard it with the appropriate ...`Free` function.

7.1.5.1 Function Reference

PmtaConnAlloc

Allocates a new connection.

```
PmtaConn PmtaConnAlloc()
```

Return   **Value:**
new connection, or 0 if not enough memory was available.

Description:

Allocates a new connection. After allocating, invoke `PmtaConnConnect` to establish the connection to the PowerMTA server.

PmtaConnConnect

```
Warning: This function is deprecated and may be removed in future
versions of the API.  Use one of the Connect functions below instead.
```
Connects to a PowerMTA server.

```
BOOL PmtaConnConnect(PmtaConn connection, const char* server)
```

Arguments:

`connection`

  connection to establish.

`server`

  string containing the server to which to connect. Currently this must be `"local:"`.

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaConnGetLastError`.

Description:

Connects to a PowerMTA server.

PmtaConnConnectRemote

Submits a message for delivery.

```
PMTACALL(BOOL) PmtaConnConnectRemote(PmtaConn connection, const
char* server, int port);
```

Arguments:

```
connection
      connection to establish.
server
      string containing the server to which to connect.  This can be
      a host name or an IP address.  Use "127.0.0.1" for the local
      machine.
port
      to which port to connect.  Use 25 for the default SMTP port.
```

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with PmtaConnGetLastError.

Description:

Connects to a PowerMTA server. No login / authentication is done.

PmtaConnConnectRemoteAuth

Submits a message for delivery.

```
PMTACALL(BOOL) PmtaConnConnectRemoteAuth(PmtaConn connection,
                                         const char* server, int port,
                                         const char* username,
                                         const char* password);
```

Arguments:

```
connection
      connection to establish.
server
      string containing the server to which to connect.  This can be a
      host name or an IP address.
port
      to which port to connect.  Use 25 for the default SMTP port.
username
      local user name for logging in to PowerMTA.  PowerMTA must be
      configured to allow authentication.
password
      password for the given user name.
```

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained
with PmtaConnGetLastError.

Description:

Connects to a PowerMTA server using the given credentials.

PmtaConnSubmit

Submits a message for delivery.

```
BOOL PmtaConnSubmit(PmtaConn connection, PmtaMsg message)
```

Arguments:

```
connection
```
        connection through which to submit the message.
```
message
```
        message to submit.

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be
obtained with PmtaConnGetLastError.

200

Description:
Submits a message for delivery.

PmtaConnFree

Discards this connection object, freeing any resouces used by it.

```
void PmtaConnFree(PmtaConn connection)
```

Arguments:

```
connection
```
connection to discard.

Description:
Discards this connection object, freeing any resouces used by it. This automatically closes the connection to the server, if any.

PmtaConnGetLastError

Returns the last error occurred on this connection.

```
const char* PmtaConnGetLastError(PmtaConn connection)
```

Arguments:

```
connection
```
connection to query for an error.

Return Value:
string containing the reason for failure.

Description:
Returns the last error occurred on this connection. This function should only be called immediately after an error occurred.

PmtaConnGetLastErrorType

Returns the type of the last error occurred.

```
int PmtaConnGetLastErrorType(PmtaConn connection)
```

Arguments:

```
connection
```
　　　　connection to query for an error.

Return Value:

type of the last error occurred, which should normally be one of the PmtaApiERROR codes defined in `PmtaApi.h`.

Description:

Returns the type of the last error occurred. This function should only be called immediately after an error occurred.

PmtaMsgAlloc

Allocates a new message.

```
PmtaMsg PmtaMsgAlloc()
```

Return Value:

New message, or 0 if not enough memory was available.

Description:

Allocates a new message. Once allocated, the new message must be initialized with `PmtaMsgInit`. Once done with the message, invoke `PmtaMsgFree` to free its resources.

PmtaMsgInit

Initializes a new message.

```
BOOL PmtaMsgInit(PmtaMsg message, const char* originator)
```

Arguments:

```
message
```
　　　　Newly allocated message to initialize.
```
originator
```
　　　　Address for the new message's originator.

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:

Initializes a new message.

202

PmtaMsgFree

Frees memory allocated for this message.

```
void PmtaMsgFree(PmtaMsg message)
```

Arguments:

```
message
```
      Message to free.

Description:

Frees memory allocated for this message.

PmtaMsgSetVerp

Sets whether this is a VERP message.

```
BOOL PmtaMsgSetVerp(PmtaMsg message, BOOL isVerp)
```

Arguments:

```
message
```
      Message to set.
```
isVerp
```
      `TRUE` if the message should be VERP.

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:

Sets whether this is a VERP message. If you enable VERP, PowerMTA will encode both the originator and recipient addresses in the originator address used for delivering the message. This may make tracking any bounces easier, since you can always derive the original addresses from the address where the bounce is sent.

PmtaMsgSetReturnType

Sets the type of message return requested in delivery reports.

```
BOOL PmtaMsgSetReturnType(PmtaMsg message, PmtaMsgRETURN type)
```

Arguments:

203

`message`
>	Message to set.

`type`
>	This should be one of:

PmtaMsgRETURN_HEADERS	(default) return only this message's headers
PmtaMsgRETURN_FULL	return the full message headers and body

Return Value:
`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:
Sets the type of message return requested in delivery reports.

PmtaMsgSetEnvelopeId

Sets this message's envelope ID.

```
BOOL PmtaMsgSetEnvelopeId(PmtaMsg message, const char* envelopeId)
```

Arguments:

`message`
>	Message to set.

`envelopeId`
>	Envelope ID for this message.

Return Value:
`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:
Sets this message's envelope ID.

PmtaMsgSetVirtualMta

Selects the VirtualMTA to use for this message.

```
BOOL PmtaMsgSetVirtualMta(PmtaMsg message, const char* virtualMta)
```

Arguments:

`message`

204

Message to set.

`virtualMta`
name of the VirtualMTA to use.

Return Value:
TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:
Selects the VirtualMTA to use for this message.

PmtaMsgSetJobId

Sets the job id for this message.

```
BOOL PmtaMsgSetJobId(PmtaMsg message, const char* jobId)
```

Arguments:

`message`
Message to set.
`jobId`
id of the job, printable, non-white space characters only.

Return Value:
TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:
Sets the job id for this message. This tags the message as belonging to the specified job.

PmtaMsgAddRecipient

Adds a recipient to this message.

```
BOOL PmtaMsgAddRecipient(PmtaMsg message, PmtaRcpt recipient)
```

Arguments:

`message`
Message to set.
`recipient`
Recipient to add.

205

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:

Adds a recipient to this message. Do not make any changes to a recipient after it has been added to the message. Once added, a recipient should be freed with `PmtaRcptFree`.

PmtaMsgSetEncoding

Sets the kind of encoding to perform on data.

```
BOOL PmtaMsgSetEncoding(PmtaMsg message, PmtaMsgENCODING encoding)
```

Arguments:

`message`
> Message to set.

`encoding`
> Data encoding desired. This should be either `PmtaMsgENCODING_7BIT`, `PmtaMsgENCODING_8BIT` or `PmtaMsgENCODING_BASE64`.

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:

Sets the kind of encoding to perform on data. The encoding applies to all thereafter added data. The actual processing performed depends on the encoding selected: `PmtaMsgENCODING_7BIT` and `PmtaMsgENCODING_8BIT` both select the *identity* encoding, i.e., no transformation is performed on the data, except that any lines terminated with LF alone are converted to CRLF. In the case of 7-bit encoding, it is your responsibility to ensure that all bytes have their high bit off. `PmtaMsgENCODING_BASE64` specifies that any data added shall be encoded on the fly using base-64 encoding. No transformation is performed on the data prior to encoding, making it possible to transmit binary content. Note, however, that to effectively transmit binary data, you must also provide the appropriate MIME headers. In fact, the selected encoding should, in general, agree with the `Content-Transfer-Encoding` header you specify. The default (if this function is never invoked) is `PmtaMsgENCODING_7BIT`.

206

PmtaMsgBeginPart

Starts the next mailmerge part which will have the given number.

```
BOOL PmtaMsgBeginPart(PmtaMsg message, int partNum)
```

Arguments:

message
>   Message to set.

partNum
>   positive integer for the part number. No part number may be specified more than once.

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with PmtaMsgGetLastError.

Description:

Starts the next mailmerge part which will have the given number. Note that part 1 is started automatically, so the first settable number is 2.

PmtaMsgAddData

Adds (appends) data to this message.

```
BOOL PmtaMsgAddData(PmtaMsg message, const char* data, int length)
```

Arguments:

message
>   Message to which to add.

data
>   Data to add.

length
>   Length of the data block.

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with PmtaMsgGetLastError.

Description:

Adds (appends) data to this message. Use this function to give this message both its headers and a body. How the data you add is handled depends on the kind of encoding selected (see PmtaMsgSetEncoding). All the data is entered as-is, with no mailmerge

207

variable substitution being performed. To add mailmerge data, use `PmtaMsgAddMergeData`.

PmtaMsgAddMergeData

Adds (appends) mailmerge data to this message.

```
BOOL PmtaMsgAddMergeData(PmtaMsg message, const char* data, int length)
```

Arguments:

`message`
    Message to which to add.
`data`
    Mailmerge data to add.
`length`
    Length of the data block.

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:

Adds (appends) mailmerge data to this message. Use this function to give this message both its headers and a body. How the data you add is handled depends on the kind of encoding selected (see `PmtaMsgSetEncoding`). Currently only 7-bit and 8-bit encodings are supported. If any mailmerge variables are included in the data, they will be substituted by their respective values.

PmtaMsgAddString

Adds (appends) a string to this message.

```
BOOL PmtaMsgAddString(PmtaMsg message, const char* str)
```

Arguments:

`message`
    Message to which to add.
`str`
    Null-terminated string to add.

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be

208

obtained with `PmtaMsgGetLastError`.

Description:
Adds (appends) a string to this message. This function is identical to `PmtaMsgAddData`, except that it expects a '0'-terminated string and computes the length of the data automatically. It is provided as a convenience function only.

PmtaMsgAddMergeString

Adds (appends) a mailmerge string to this message.

```
BOOL PmtaMsgAddMergeString(PmtaMsg message, const char* str)
```

Arguments:

`message`
      Message to which to add.
`str`
      Null-terminated string to add.

Return Value:
`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:
Adds (appends) a mailmerge string to this message. This function is identical to `PmtaMsgAddMergeData`, except that it expects a '0'-terminated string and computes the length of the data automatically. It is provided as a convenience function only.

PmtaMsgAddDateHeader

Adds (appends) a `Date:` header to this message.

```
BOOL PmtaMsgAddDateHeader(PmtaMsg message)
```

Arguments:

`message`
      Message to which to add.

Return Value:
`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaMsgGetLastError`.

Description:

Adds (appends) a `Date:` header to this message. This function appends a date header line to the message. It is provided as a convenience function only.

PmtaMsgGetLastError

Returns the last error occurred on this message.

```
const char* PmtaMsgGetLastError(PmtaMsg message)
```

Arguments:

`message`
        Message to query for an error.

Return Value:

String containing the reason for failure.

Description:

Returns the last error occurred on this message. This function should only be called immediately after an error occurred.

PmtaMsgGetLastErrorType

Returns the type of the last error occurred.

```
int PmtaMsgGetLastErrorType(PmtaMsg message)
```

Arguments:

`message`
        Message to query for an error.

Return Value:

Type of the last error occurred, which should normally be one of the PmtaApiERROR codes defined in `PmtaApi.h`.

Description:

Returns the type of the last error occurred. This function should only be called immediately after an error occurred.

PmtaRcptAlloc

Allocates a new recipient.

```
PmtaRcpt PmtaRcptAlloc()
```

Return Value:

New recipient, or 0 if not enough memory was available.

Description:

Allocates a new recipient. Once allocated, the new recipient must be initialized with `PmtaRcptInit`. When no longer needed (e.g., after adding the recipient to a message), its resources should be freed by calling `PmtaRcptFree`.

PmtaRcptInit

Initializes a new recipient.

```
BOOL PmtaRcptInit(PmtaRcpt recipient, const char* address)
```

Arguments:

`recipient`
>   Newly allocated recipient to initialize.

`address`
>   e-mail address for the recipient.

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaRcptGetLastError`.

Description:

Initializes a new recipient.

PmtaRcptFree

Frees storage allocated by this recipient.

```
void PmtaRcptFree(PmtaRcpt recipient)
```

Arguments:

`recipient`
>   Recipient to free.

Description:

Frees storage allocated by this recipient. To avoid memory leaks, this function should

211

be called on every recipient allocated, whether it has been added to a message or not.

PmtaRcptSetNotify

Sets the kind of notification (report) desired for this recipient.

```
BOOL PmtaRcptSetNotify(PmtaRcpt recipient, int notifyWhen)
```

Arguments:

`recipient`
Recipient to set

`notifyWhen`
This should be either `PmtaRcptNOTIFY_NEVER` to indicate that no notification is desired, or one or more of the following flags (joined by a bitwise OR):

PmtaRcptNOTIFY_SUCCESS	notify in case the delivery is successful
PmtaRcptNOTIFY_FAILURE	notify in case of a delivery failure
PmtaRcptNOTIFY_DELAY	notify in case of a delay in the delivery

Return Value:

`TRUE` in case of success, `FALSE` in case of failure. The cause of the error can be obtained with `PmtaRcptGetLastError`.

Description:

Sets the kind of notification (report) desired for this recipient. By default, notification is only requested upon delivery failure. Note that while PowerMTA does not support notification for delivery delays, the corresponding `DSN` flag is passed on to the receiving mailer, which may support it.

PmtaRcptDefineVariable

Defines a new mailmerge variable for this recipient.

```
BOOL PmtaRcptDefineVariable(PmtaRcpt recipient, const char* name, const char* value)
```

Arguments:

`recipient`
recipient whose variable is being defined.

`name`
the variable's name.

`value`
the variable's value.

212

Return Value:

TRUE in case of success, FALSE in case of failure. The cause of the error can be obtained with
PmtaRcptGetLastError.

Description:

Defines a new mailmerge variable for this recipient. PowerMTA will substitute the variable for its value during delivery of a mailmerge message.

Carriage returns and line feeds may be added to a variable with \r or \n respectively (Assuming that rcpt is of type PmtaRcpt and is properly initialized).

```
if (!PmtaRcptDefineVariable(rcpt, "cr", "A\rB")) {      // one linebreak
    fprintf(stderr, "Error defining cr variable: %s\n",
                    PmtaRcptGetLastError(rcpt));
    return 1;
}
if (!PmtaRcptDefineVariable(rcpt, "lf", "A\nB")) {      // one linebreak
    fprintf(stderr, "Error defining lf variable: %s\n",
                    PmtaRcptGetLastError(rcpt));
    return 1;
}
if (!PmtaRcptDefineVariable(rcpt, "crlf", "A\r\nB")) {  // one linebreak
    fprintf(stderr, "Error defining crlf variable: %s\n",
                    PmtaRcptGetLastError(rcpt));
    return 1;
}

if (!PmtaRcptDefineVariable(rcpt, "literalCrlf", "A\\r\\nB")) {  // no linebreaks
    fprintf(stderr, "Error defining literalCrlf variable: %s\n",
                    PmtaRcptGetLastError(rcpt));
    return 1;
}
```

PmtaRcptGetLastError

Returns the last error occurred on this recipient.

```
const char* PmtaRcptGetLastError(PmtaRcpt recipient)
```

Arguments:

recipient
        Recipient to query for an error.

Return Value:

String containing the reason for failure.

Description:

213

Returns the last error occurred on this recipient. This function should only be called immediately after an error occurred.

PmtaRcptGetLastErrorType

Returns the type of the last error occurred.

```
int PmtaRcptGetLastErrorType(PmtaRcpt recipient)
```

Arguments:

```
recipient
```
        Recipient to query for an error.

Return Value:
Type of the last error occurred, which should normally be one of the PmtaApiERROR codes defined in `PmtaApi.h`.

Description:
Returns the type of the last error occurred. This function should only be called immediately after an error occurred.

7.1.6 .NET Submission API

Documentation for the .NET APIs is provided in doc folder.

On Windows, it can be found in the `api\dotnet` subdirectory of where PowerMTA was installed.

On Linux it can be found at https://download.port25.com

7.2 Migrating from the Old Submission APIs

In PowerMTA v3.5 we redesigned our submission APIs, incorporating customer requests, facilitating planned enhancements, as well as tidying up the design a little to make it more intuitive and easier to use. While the differences between old and new APIs are not huge, this section is intended to facilitate the migration to the new APIs.

7.2.1 Overview of Differences Between Old and New APIs

The most notable difference is, that all APIs now support remote feeding. To use the new APIs, you will need to recompile your existing APIs using the new libraries. In addition to the remote feeing, PowerMTA now uses a modified version of SMTP. This modified version is our MAIL MERGE feature set and can help with increasing the feeding rate into PowerMTA. For more information on MAIL MERGE, see Section 9.

7.3 Accounting APIs

No longer supported.

7.4 Pickup Directory

PowerMTA can be instructed to pick up message files from a directory. To submit a message, you write it to a file, formatting it according to Internet e-mail standards but preceding it with envelope information in special headers: `x-sender`, `x-receiver` and, optionally, `x-envid` and `x-job`. You then move this file to the pickup directory, from where PowerMTA reads it into its message spool and performs delivery. It is recommend to add no more than 5000 recipients per pickup file.

The `x-sender` header should be in the first line in the file, followed by one or more `x-receiver` headers, each in its own line. You can also optionally set the DSN envelope ID and the job ID by adding the `x-envid` and `x-job` headers respectively. The `x-sender` and `x-receiver` headers must contain the e-mail addresses only, with *no* free form name, no "<>"s, etc., like in:

```
x-sender: me@over.here
x-receiver: you@over.there
x-receiver: someone.else@some.other.place
x-envid: me-0001
x-job: love-2003-01
From: Myself Personally <me@over.here>
To: You <you@over.there>
Date: Thu, 6 Jul 2000 07:45:43 -0400
Subject: I love it

Hey, did you know that ...
```

Note that in the example above, `someone.else@some.other.place` is BCC'ed, i.e., blind copied: he or she receives a copy of the message but is not included in the recipient headers.

PowerMTA's pickup directory also supports the `x-sender-override` header, which allows you to override the contents of `x-sender`. If `x-sender-override` is present, its contents override the sender specified in `x-sender`, if any. This is useful when you wish to customize the message sender despite the software you're using to create the messages not allowing for that. Typically that software will still allow you to pass additional headers, such as `x-sender-override`.

Other than reading the sender and receiver e-mail address headers and removing them from the message, no other header or body processing takes place. You are responsible for properly formatting the message.

Instead of creating the file elsewhere and then moving to the pickup directory, it is also possible to create the file directly in it. However, since PowerMTA must repeatedly attempt to lock the file for exclusive access, it is less efficient to do so.

To configure PowerMTA for a pickup directory, specify the pickup directory as described in Section 3.2.13. You must also specify a directory as the "bad mail" directory, to which messages are moved if there is a permanent error processing the message.

7.4.1 BSMTP files

Starting with PowerMTA v3.0, the pickup directory now also supports BSMTP (batched SMTP) files. A BSMTP file is simply a file that contains all of the SMTP commands necessary for submitting a message batched together in one text file, as in the example below:

```
MAIL FROM:<me@over.here>
RCPT TO:<you@over.there>
DATA
From: Myself Personally <me@over.here>
To: You <you@over.there>
Date: Thu, 6 Jul 2000 07:45:43 -0400
Subject: I love it

Hey, did you know that ...
.
```

The pickup directory supports the following SMTP commands: `HELO`, `EHLO`, `MAIL`, `RCPT` and `DATA`. Additionally, it also supports PowerMTA's extended SMTP mailmerge commands (`XMRG`, `XDFN`, `XPRT` and `XACK`), allowing for both mailmerge and non-mailmerge messages to be created.

To be recognized as such, BSMTP files need not be marked in any special way, other than the first command be one of: `HELO`, `EHLO`, `MAIL`, `XACK` or `XACK`.

Like in traditional pickup files, only one message may be submitted per file. However, since mailmerge is supported, the message submitted may actually be a template used for many recipients. See [Section 9.5.3](#) for examples on using the pickup directory for submitting mailmerge messages.

If any commands cannot be successfully processed, the entire file is rejected and moved to the "bad file" directory. For certain recipient address related errors, however, you have the option of requesting that PowerMTA bounce the recipient rather than rejecting the file during submission. This is activated by disabling recipient acknowledgements, using PowerMTA's XACK SMTP extension, like in the example below:

```
XACK OFF
MAIL FROM:<me@over.here>
RCPT TO:<you@over.there>
RCPT TO:<notyou@not..here>
DATA
From: Myself Personally <me@over.here>
To: (recipients omitted)
Date: Thu, 6 Jul 2000 07:45:43 -0400
Subject: I love it

Hey, did you know that ...
.
```

In the example above, despite the error in the second recipient's domain, the file would not be rejected, but a bounce would be sent to the address in the MAIL FROM.

[Section 9.5.2](#) describes PowerMTA's mailmerge extensions; look at [Section 9.5.4](#) for the XACK extension.

7.5 Pipe Delivery

PowerMTA can be configured to deliver e-mail to another program on the same machine. It does so by starting the program for each new recipient and writing the message to the program's standard input. The mechanism used to pass on the message is called "pipe", so this feature is named "pipe delivery".

****Important note****
Any pipe program should work outside of PowerMTA by simply feeding data into it via an echo command or orther such standard out (e.g. cat-ing a file). If a program does not work outside of PowerMTA, it will not work inside of PowerMTA. If the program works outside of PowerMTA, but not inside PowerMTA then the issue is mostly a permissions issue in the directory/file to which PowerMTA is trying to write.

To use pipe delivery, you add domain definitions for one or more domains, specifying a `type` of `pipe` and defining the command to run for starting the pipe receiver program. For example:

```
<domain bounces.company.com>
    type pipe
    command "c:\\path\\bounceproc.exe --envid $envid"
</domain>
```

Note that since the backslash character is also used for escaping other characters (such as spaces, quotes and the dollar sign), you need to specify it twice where you want just one to appear. On Windows, you can also use the (forward) slash character to specify paths, so in the example above it could have been `c:/path/bounceproc.exe`.

Before starting the program, PowerMTA substitutes any macros found in the command line (such as `$envid` in the example above), allowing you to pass on additional message-related parameters to the pipe receiver program. The complete list of macros supported and other configuration directives for pipe domains are described in Section 3.2.12.

7.5.1 Delivering mail to local users

PowerMTA's pipe delivery can be used to deliver e-mail to local users. The special `local` domain name makes this simple to configure by concentrating all e-mail destned to recipients on any of the local domains. PowerMTA will use the hostname of the OS, or and host-names configured in PowerMTA, to determine if an email is local. On Unix systems, it is often sufficient to specify `procmail` as the command to execute for pipe delivery:

```
<domain local>
    type pipe
    command "/usr/bin/procmail -t -Y -d $user"
</domain>
```

Note, however, that PowerMTA does not currently process the `/etc/aliases` or `.forward` files.

Alternatively, the same could be done be explicitly specifying the domain name as in this example:

```
<domain bounces.yourdomain.com>
    type pipe
    command "/usr/bin/procmail -t -Y -d $user"
</domain>
```

The same is also available on Windows, but you will need to provide a pipe delivery processor that can write the messages in a way your user agents (i.e., e-mail clients) understand.

7.5.2 Pipe Delivery Programs Included With PowerMTA

PowerMTA comes with two (fully functional) sample programs to be used with the new pipe delivery functionality; `appendtofile`, which appends messages to a file, and `newfile`, which writes each message to a separate (new) file.

- `appendtofile` appends the message and / or arguments passed to it to the file specified. The syntax for `appendtofile` is:

```
appendtofile [--nomsg] filename [text]
```

- where *filename* is the name of the file to which to append the data and *text* is an optional text or a supported macro to write in front of each message. If the first command line argument given to `appendtofile` is `--nomsg` (or `/nomsg`), it will just apppend the text or the macro's value to the file. You can thus choose to either save the complete message to the file or just the summary data passed on the command line, such as only the recipient address of the message (when using the `$to` macro).
- `newfile` writes each message to a separate (new) file. The syntax for `newfile` is simply:

```
newfile dirname
```

- where *dirname* is the directory where the message file should be created. If the given directory does not yet exist, `newfile` automatically creates it. Each message file name is automatically generated by `newfile`, and will have a '.msg' extension.

For greatest flexibility, you can use the supported macro substitutions described in Section 3.2.12 to automatically pass message-dependent file names and directory names. For example, `appendtofile c:/pipe/$domain` in the `command` directive will append messages to the file named after the domain of the recipient address, given the presence of the `$domain` macro. A message sent to `test@port25.com` would be appended to a file named `port25.com` in the directory `c:\pipe`, while a message sent to `test@powermta.com` would be appended to a file named `powermta.com` in the same directory.

In the typical scenario where the pipe delivery applications are used to store and process bounces, using the `$domain` macro results in files named after the originator / SMTP `MAIL FROM` domain you used in your original messages, since the recipient address of a bounce report is the original `MAIL FROM` address used.

For example, if you used a `MAIL FROM` domain of `bounces5.port25.com` for all of your messages for customer or campaign ID #5, the file `bounces5.port25.com` would contain the bounce reports for the messages of this customer or campaign. The configuration for this would be;

```
<domain bounces5.port25.com>
    type pipe
    command "c:/pmta/bin/appendtofile.exe c:/pipe/$domain"
</domain>
```

Alternatively, with the `--nomsg` and passing the `$to` macro to `appendtofile` as well, like in

```
<domain bounces5.port25.com>
    type pipe
    command "c:/pmta/bin/appendtofile.exe --nomsg c:/pipe/$domain $to"
</domain>
```

only the recipients would be written to a file (given the `$to`), which is named after the domain of the recipients (given the `$domain`), `port25.com`.

With `newfile`, if you had a configuration like

```
<domain bounces5.port25.com>
    type pipe
    command "c:/pmta/bin/newfile.exe c:/pipe/$domain"
</domain>
```

`newfile` will create subdirectories named after the recipients' domains, and only create message files within the appropriate subdirectories. Using the example above with the domain `bounces5.port25.com` as the domain of your originator address, when bounces come in, the directory created would be `c:\pipe\bounces5.port25.com`, and which would contain only files / message received for the recipientrs of that domain. If you had PowerMTA handling inbound mail for 50 separate domains, `newfile` would create 50 separate subdirectories, each named for a different domain, and containing only messages for that domain.

7.5.3 Accessing Files Created By The Sample Applications

Since, in principle, new messages may be delivered at any time through the sample applications, to avoid loss of data special care must be taken when accessing the files these applications create.

`newfile` solves concurrency conflicts by using a naming convention: it initially creates files with a `.tmp` suffix ("extension") and when done writing to them, renames them to `.msg`. This makes it very easy to access the files: you can freely access all the `.msg` files, while leaving any `.tmp` files alone.

Doing the same for `appendtofile` is a little more challenging, since it writes data to the same file and due to the differences in file lock handling across the PowerMTA platforms. To allow access to `appendtofile`-generated files in a simple and uniform way, Port25 provides a program named `pmtagetfile`, which moves the contents of a given file.

To use `pmtagetfile`, you simply run it passing the name of the file to which `appendtofile` writes and the new destination file name on the command line. For example,

```
pmtagetfile c:/pmta-pipe/port25.com.txt c:/tmp/port25.com.txt
```

moves the file `c:\pmta-pipe\port25.com.txt` to `c:\tmp\port25.com.txt`.

Once in the new directory, the file is out of `appendtofile`'s reach, and you can safely process it as required. Note that the `pmtagetfile` program will fail if the first file listed does not exist (nothing to copy from) or if the second file already exists (preventing accidental overwriting of previous data). Also, `pmtagetfile` does not work across file system boundaries, i.e., the the new file must be on the same file system or drive (on Unix and Windows systems, respectively). To move a file to a different file system, use `pmtagetfile` to first move it away from `appendtofile` to a safe place on the same file system, and then move that file with your usual tools (Explorer, shell commands, etc.) to the desired location.

Please note that `pmtagetfile` is only intended for use with files written by `appendtofile` or with files written to by File Delivery.

7.5.4 Writing Your Own Pipe Delivery Programs

For convenience and flexibility, the sample pipe delivery programs detailed above are also provided in source code, allowing you to modify or customize as needed.

The program used for pipe delivery should terminate with certain specific exit codes. On Unix systems, these are defined in `/usr/include/sysexits.h`. On Windows, a corresponding file is included in the `api\include` directory. Depending on the exit code, PowerMTA will consider the delivery to have succeeded or to have temporarily or permanently failed and generate any DSN delivery reports as appropriate:

exit code	action	DSN status
EX_OK	delete	2.0.0 (success)
EX_USAGE	bounce	5.5.4 (invalid command arguments)
EX_NOUSER	bounce	5.1.1 (bad destination mailbox address)
EX_NOHOST	bounce	5.1.2 (bad destination system address)
EX_CANTCREAT	bounce	5.2.0 (mailbox-related failure)
EX_UNAVAILABLE	bounce	5.3.0 (mail system-related failure)
EX_SOFTWARE	bounce	5.3.0 (mail system-related failure)
EX_OSFILE	bounce	5.3.0 (mail system-related failure)
EX_NOINPUT	bounce	5.3.0 (mail system-related failure)
EX_OSERR	retry	4.3.0 (mail system-related failure)
EX_IOERR	retry	4.3.0 (mail system-related failure)
EX_TEMPFAIL	retry	4.0.0 (transient failure)
EX_DATAERR	bounce	5.5.2 (syntax error)
EX_PROTOCOL	bounce	5.5.0 (protocol-related failure)
EX_CONFIG	bounce	5.3.5 (system incorrectly configured)
EX_NOPERM	bounce	5.7.1 (delivery not authorized)

If you use `appendtofile` but for some reason would like to replace `pmtagetfile`, please look into `pmtagetfile`'s sources first to better understand how it works to avoid file locking conflicts.

8. VirtualMTA Support

8.1 Overview

Increasing volumes of e-mail have prompted many ISPs and e-mail administrators to proactively filter or block incoming e-mail. While their intentions are honest, this also results in the blocking of legitimate mail for sites running on or delivering through a shared mail gateway. Traditionally, all mail sent from a shared MTA or gateway is sent from same host name and IP addresses. Since ISPs and e-mail administrators tend to initially key on the source IP address for blocking incoming mail, more aggressive senders, or simply higher volume customers delivering through a shared MTA will almost certainly put all others at risk of being blocked.

To maximize deliverability and minimize collateral blocking of legitimate email on shared gateways, Port25 has developed *VirtualMTA* technology within PowerMTA. Simply put, this functionality allows sites to define and control both the source IP address and host name for each message. This allows for creating VirtualMTAs for separate and specific campaigns, customers or departments, while still running just one instance of PowerMTA.

8.2 VirtualMTA Definitions

VirtualMTAs are defined via the new `<virtual-mta>` directive. Defining a VirtualMTA entails creating a `<virtual-mta>` group with the desired alias and listing its host name and SMTP source IP(s). For example, if you wanted to create a VirtualMTA named `mta1`, you would simply add in the configuration file:

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 mta1.port25.com
</virtual-mta>
```

Changes to the VirtualMTA definitions are reloadable via the `pmta reload` command and thus do not require a restart.

Note that if you leave out the `smtp-source-host` directive in the definition, PowerMTA will use the corresponding global settings for that VirtualMTA. Since that may or may not be the desired result, Port25 always recommends that you specify those directives within each VirtualMTA definition. For example, you could use the desired host name as the VirtualMTA alias, but you would still need to specify the `host-name` directive to ensure that the right host name is used:

```
<virtual-mta mta2.port25.com>
    smtp-source-host 2.3.4.5 mta2.port25.com
</virtual-mta>
```

Lastly, if more than one `smtp-source-host` address is listed in a definition, PowerMTA will use the various source IPs in round robin fashion for connections from that VirtualMTA:

```
<virtual-mta mta3>
    smtp-source-host 1.2.3.4 mta4.port25.com
    smtp-source-host 2.3.4.5 mta5.port25.com
    smtp-source-host 3.4.5.6 mta6.port25.com
</virtual-mta>
```

VirtualMTA directives only apply when the VirtualMTA has been selected as described in Section 8.4. Otherwise, they are simply ignored. Note that the `host-name` and `smtp-source-host` directives are also global system directives, detailed in Chapter 3.

8.3 VirtualMTA Pools

PowerMTA supports the defining and selecting of VirtualMTA pools per message or campaign as well, which are basically multiple VirtualMTAs grouped in a pool. When a pool is defined and selected for a campaign, PowerMTA will use the VirtualMTAs listed in round robin fashion for connections from that pool.

VirtualMTA pools are defined via the new `<virtual-mta-pool>` directive. Defining a pool entails creating a `<virtual-mta-pool>` group with the desired alias and listing the VirtualMTAs that you want belonging to that pool. For example, if you wanted to create a VirtualMTA pool named `pond`, you would first define the specific VirtualMTAs to be part of `pond`, as described in Section 8.2, and then subsequently list each of the VirtualMTAs that is to be included using the `virtual-mta` directive:

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 mta1.port25.com
</virtual-mta>

<virtual-mta mta2>
    smtp-source-host 2.3.4.5 mta2.port25.com
</virtual-mta>

<virtual-mta-pool pond>
    virtual-mta mta1
    virtual-mta mta2
</virtual-mta-pool>
```

Note that PowerMTA will not accept a VirtualMTA pool definition if the pool name equals a previously defined VirtualMTA name, given that you select a VirtualMTA pool and/or VirtualMTA via the same headers and function calls, as detailed in Section 8.4.

A <domain> may be defined inside of a <virtual-mta-pool>. If this is done, the defined directives will be inherited by any VirtualMTA that is part of the pool. If a conflicting directive is defined directly in the given VirtualMTA, the directive in the VirtualMTA overrides the one defined in the <virtual-mta-pool>.

The following directives are also supported:

- cold-virtual-mta

- domain-key

- max-smtp-out

- max-smtp-msg-rate

- include-headers-from

8.4 Selecting a VirtualMTA or VirtualMTA Pool

PowerMTA supports various methods of selecting a VirtualMTA (or pool), depending on your submission method:

- Submission APIs:
    - by defining the *vmta mailmerge variable in mailmerge messages;
    - by using the setVirtualMta or equivalent method (see Chapter 7 for details);
- Pickup directory submission:
    - by defining the *vmta mailmerge variable in mailmerge messages;
    - by pattern matching based on the MAIL FROM, RCPT TO, or a header.
- SMTP submission:
    - by defining the *vmta mailmerge variable in mailmerge messages;
    - by including a X-virtual-MTA header in the message (see Section 8.4.1 below for details);
    - by regular expression matching of the SMTP MAIL FROM or RCPT TO addresses, or a message header; see Section 3.2.16 and the pattern-list directive in Section 3.2.4 as well as Section 8.4.2 below for details);
    - by configuring a default VirtualMTA on a <source> directive selected by the IP address and port in the PowerMTA machine that you connect to (see the smtp-listener directive in Section 3.2.4 as well as Section 8.4.3 below for details);

o by configuring a default VirtualMTA on a `<source>` directive selected by the source IP address (see the `default-virtual-mta` directive in [Section 3.2.4](#) as well as [Section 8.4.4](#) below for details);

The selection methods above are listed in the order of precedence within each submission methods.

Using mailmerge may effect how VirtualMTA selection is done. When using mailmerge and the API or SMTP, the x-virtual-mta will apply to the entire mailmerge job, not individual recipients (in most cases using the first defined x-virtual-mta).

When using mailmerge and pattern lists, VirtualMTA selection will be for each recipient when matching the RCPT TO, and will only be for each recipient for the MAIL FROM when using the *from mailmerge variable. When not using the *from, or when using header matching, *all* recipients in the mailmerge job will use the same VirtualMTA (in most cases the first matched pattern).

Note that one selects a VirtualMTA pool using the same exact header, directives or function call, but using the pool name instead of the VirtualMTA name.

8.4.1 Selecting a VirtualMTA with a `X-virtual-MTA` Header

To select a VirtualMTA with the `X-virtual-MTA` header, you need to:

1. define the VirtualMTA(s) or VirtualMTA pool(s) in the configuration file using the `<virtual-mta>` directive;
2. define the specific IP address or IP address range for the source of the messages that it is applicable using the expanded `<source>` directive;
3. define the ability for PowerMTA to process the `x-virtual-mta` header via the `process-x-virtual-mta` directive, within the `<source>` directive defined in step 2;

After setting up the configuration file properly, you simply need to then add the `x-virtual-mta` header to your message specifying the VirtualMTA's alias, like in

```
x-virtual-mta: yyyyy
```

where *yyyyy* is the VirtualMTA's alias in the configuration file. Note that PowerMTA will parse and remove the header before ultimate delivery, so it will not be seen by the recipient's mail gateway or the recipient.

For example, assuming that you had the following VirtualMTA definition in your configuration file:

226

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 mta1.port25.com
</virtual-mta>
```

and you wanted PowerMTA to process the `x-virtual-mta` header from mail only coming from the local IP address 127.0.0.1 for selecting a VirtualMTA, you would need to have at a minimum the following in your configuration file;

```
<source 127.0.0.1>
    process-x-virtual-mta yes   # allow selection of a VirtualMTA
    always-allow-relaying yes   # allow feeding from 127.0.0.1
</source>
```

With this, any messages submitted from the local IP address 127.0.0.1 that contained the header `x-virtual-mta: mta1` would have PowerMTA using VirtualMTA `mta1` and its parameters when making connections to deliver these messages. One can also select a VirtualMTA pool to be used for messages in the same manner.

The entire message (in this example, submitted through the pickup directory) could thus read:

```
x-sender: me@over.here
x-envid: me-0001
x-receiver: you@over.there
x-virtual-mta: mta1
From: Myself Personally <me@over.here>
To: You <you@over.there>
Date: Thu, 13 Mar 2003 07:45:43 -0400
Subject: PowerMTA is the best

Hey, this new VirtualMTA technology really improves delivery
rates!
```

8.4.2 Selecting a VirtualMTA by Regular Expression Matching

Sites can also have PowerMTA use a VirtualMTA or VirtualMTA pool for specific messages by way of regular expression pattern matching of the SMTP `MAIL FROM` address, the SMTP `RCPT TO` address, or a header, based on the IP address of the source of the message.

To select a VirtualMTA or VirtualMTA pool based on regular expression pattern matching, you

1. define the VirtualMTAs or VirtualMTA pools in the configuration file using the `<virtual-mta>` directive;

227

2. define the alias for the regular expression pattern list using the `<pattern-list>` directive;
3. define the actual regular expression pattern to use for matching the SMTP MAIL FROM, the RCPT TO address, or a header as well as subsequent VirtualMTA or VirtualMTA pool to use when a match occurs, via the `mail-from`, `rcpt-to`, and `header` directives. These directives are defined within the `<pattern-list>` definition created in step 2;
4. define the specific IP address or IP adress range for the source of the messages that is applicable within the expanded `<source>` directive;
5. define the `pattern-list` directive using the corresponding alias defined in step 2 within the `<source>` directive defined in step 4. Conditional pattern matching as defined in <u>Section 3.2.16</u> may also be used.

For example, assuming that you had the following VirtualMTA definitions in your configuration file:

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 mta1.port25.com
</virtual-mta>

<virtual-mta mta2>
    smtp-source-host 2.3.4.5 mta2.port25.com
</virtual-mta>
```

and you wanted PowerMTA to match the patterns "abc" and "xyz" at the beginning of the MAIL FROM address in order to select the appropriate VirtualMTA to use for each, you would need to have at a minimum the following in your configuration file:

```
<pattern-list group2>
    mail-from /^abc/ virtual-mta=mta1
    mail-from /^xyz/ virtual-mta=mta2
</pattern-list>

<source 127.0.0.1>
    pattern-list group2          # this selects the pattern list for messages from this IP
    always-allow-relaying yes   # allow feeding from 127.0.0.1
</source>
```

With this, any messages submitted from the local IP address 127.0.0.1 would have PowerMTA using regular expression matching on the MAIL FROM address, and if PowerMTA found "abc" at the beginning of the address (for example abc-2890u8@bounces.port25.com), VirtualMTA mta1 and its parameters would be used when making connections to deliver these messages. For messages submitted that had a MAIL FROM starting with "xyz", for example xyz-ljwer@bounces.port25.com, VirtualMTA mta2 and its parameters would instead be used when making connections to deliver these messages.

To additionally select some other VirtualMTA for recipients at `yahoo.com`, you could add a `rcpt-to` directive as below:

228

```
<pattern-list group2>
    mail-from /^abc/ virtual-mta=mta1
    mail-from /^xyz/ virtual-mta=mta2
    rcpt-to   /@yahoo.com$/ virtual-mta=mta3
</pattern-list>

...
```

Per the simple examples above, you can see that PowerMTA can use different matching patterns for different campaigns, customers and/or mailings simultaneously, in order to use different VirtualMTAs or VirtualMTA pools for each.

More than one `mail-from`, `rcpt-to` and `header` directives are allowed within a `<pattern-list>` definition, and since the first match is used, you will want to list the most likely matches first. Only one `pattern-list` is allowed within a `<source>` group.

Conditional matching is allowed as well. For example, if you want wanted all emails that have a MAIL FROM of X, and a RCPT TO of Y, that is allowed in a pattern similar to the following:

```
<pattern-list patterns>
   mail-from /X/ <pattern-list>
       rcpt-to /Y/ virtual-mta=vmta2
       rcpt-to /Z/ virtual-mta=vmta1
       * virtual-mta=vmta2
   </pattern-list>
</pattern-list>
```

For more information on conditional pattern matching, see Section 3.2.16.

Note that simple to advanced regular expression pattern matching is supported, and which is based on Perl Compatible Regular Expressions (PCRE).

8.4.3 Selecting a VirtualMTA by the IP Address/Port Receiving Connections

For added flexibility, PowerMTA supports the ability to select a VirtualMTA based on the IP address (and TCP port number) on the PowerMTA machine to which you connect when submitting your messages.

For example, if you have 10 VirtualMTAs defined, each with a different IP address, you could select the VirtualMTA to use for each message simply by making your SMTP connection to the specific IP address defined on your PowerMTA machine for the specific VirtualMTA.

This is accomplished through "named" `<source>` tags, in conjunction with the `smtp-listener` directive. Named `<source>` tags allow you to specify directives that override those normally selected by source IP address; they apply only when referenced in the `smtp-listener` directive (and for authenticated users — see Section 10.2.2).

For example, to select VirtualMTA `vmta1` by its assigned `smtp-source-host` address `1.2.3.4` and the standard SMTP port (`25`), you would configure:

```
<virtual-mta vmta1>
    smtp-source-host 1.2.3.4 vmta1.yourdomain.com
    ...
</virtual-mta>

<source vmta1>
    default-virtual-mta vmta1
</source>

smtp-listener 1.2.3.4:25 source=vmta1
```

Multiple `smtp-listener` entries can be specified in a configuration file, each of which specifying a different IP address and/or port.

Note that if any `smtp-listener` directives are specified, all listeners must be explicitly defined: PowerMTA no longer starts a listener automatically based on the value of the the `smtp-ip` and `smtp-port` directives. When configuring your first `smtp-listener` entry, to retain the default listener you should add another `smtp-listener` directive specifying your `smtp-ip` and `smtp-port` in it. If you have no `smtp-ip` configured, use `0.0.0.0` instead, which causes PowerMTA to bind to all local IP addresses:

```
smtp-listener 0.0.0.0:25
```

Since the named source's directives apply to anyone who can connect to the listener's IP address and port, one should be careful about what is entered there. For example, `always-allow-relaying` should most probably **not** be enabled in the named source. Instead, permission for relaying, for accessing the mailmerge extensions, etc. should be based on authenticated sources (see Section 10.2.2) or on the source IP address, as in the example below:

```
<virtual-mta vmta1>
    smtp-source-host  1.2.3.4  vmta1.yourdomain.com
    ...
</virtual-mta>

smtp-listener 1.2.3.4:25 source=vmta1
smtp-listener 0.0.0.0:26

<source vmta1>
    default-virtual-mta vmta1
```

230

```
</source>

<source 10.0.0.0/8>
    allow-mailmerge yes
    always-allow-relaying yes
</source>

<source 0/0>
    allow-mailmerge no
    always-allow-relaying no
</source>
```

Above, VirtualMTA vmta1 is selected by default for any messages received through `1.2.3.4` on the PowerMTA machine. The default listener on `0.0.0.0:25` is retained through its own `smtp-listener` directive. Permission for relaying and for using the mailmerge extensions is granted to connections coming from `<source 10.0.0.0/8>`, whether they connect to `1.2.3.4` or not.

8.4.4 Selecting a VirtualMTA by the Source IP Address

For sites that are using applications that do not allow them to create custom `x-`headers within their messages, or for sites feeding PowerMTA messages from a variety of different SMTP applications and machines, PowerMTA supports the ability to select and use VirtualMTAs based only on the IP address of the source of the message.

To select a VirtualMTA or VirtualMTA pool based on the IP address of the original source of the message, you need to;

1. define the VirtualMTAs or VirtualMTA pools in the configuration file using the `<virtual-mta>` directive;
2. define the specific IP address or IP address range for the source of the messages that is applicable using the expanded `<source>` directive;
3. define the VirtualMTA to use for messages via the new `default-virtual-mta` directive within the `<source>` directive defined in step 2.

For example, assuming that you had the following VirtualMTA definition in your configuration file:

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 mta1.port25.com
</virtual-mta>
```

and you wanted only mail coming from the local IP address 127.0.0.1 to use this VirtualMTA by default for each message (without having to specify an `x-virtual-mta`

231

header in the messages), you would need to have at a minimum the following in your configuration file;

```
<source 127.0.0.1>
    default-virtual-mta   mta1  # selects a VirtualMTA/pool by source of the message
    always-allow-relaying yes   # allow feeding from 127.0.0.1
</source>
```

With this, any messages submitted from the local IP address 127.0.0.1 would have PowerMTA using VirtualMTA `mta1` and its parameters when making connections to deliver these messages. One can also select a VirtualMTA pool to be used for messages coming from the specific IP address by instead specifying a VirtualMTA pool name via the `default-virtual-mta` directive.

You can basically define as many different source IP addresses or IP address ranges for messages in the configuration file as you wish, with each using a different VirtualMTA or VirtualMTA pool.

8.4.5 Selecting a VirtualMTA by a VirtualMTA IP address

In addition to the above, a VirtualMTA can be slected based on which PowerMTA IP is used for the connection, and the corresponding VirtualMTA that is configured with that IP. To use this selection method, the source directive "`default-virtual-mta`" will need to be used with a setting of "`by-smtp-source-ip`" in place of a VirtualMTA name.

When configured in this way, on an inbound SMTP connection PowerMTA will use as the default VirtualMTA that whose "smtp-source-host" setting equals the destination IP address on the inbound connection.

Example:

```
<virtual-mta mta1>
    smtp-source-host 1.2.3.4 vmta1.port25.com
</virtual-mta>

<source 0/0>
    default-virtual-mta by-smtp-source-ip
</source>
```

In the above example, connecting to IP address 1.2.3.4 would automatically route the message through VirtualMTA "mta1". Also, during the connection from the external mailer to PowerMTA, PowerMTA would respond with the host name configured in the VirtualMTA.

8.5 Changing VirtualMTAs on the Fly

There may be times when you need to change or even cancel a VirtualMTA definition after mail has already been queued for it, so PowerMTA has been designed to handle these scenarios.

8.5.1 Changing a VirtualMTA

If you change the host name and/or source IP addresses of a VirtualMTA after e-mail has already been queued, PowerMTA will use the new information for all new connections once the configuration file has been reloaded. However, if connections were already established when the changes are loaded, the old VirtualMTA information will be used for the duration of those connections.

8.5.2 Cancelling a VirtualMTA

If you would like to cancel a VirtualMTA after messages have already been queued, you can simply delete or comment its definition out of the configuration file (and run the `pmta reload` command). PowerMTA will use the new information for all new connections once the configuration file has been reloaded. However, if connections were already established when the changes are loaded, the old VirtualMTA information will be used for the duration of those connections.

If a message selects a VirtualMTA which does not exist when PowerMTA attempts to deliver it, that message is bounced with the DSN status

```
Status: 5.3.5 (specified VirtualMTA does not exist)
```

8.6 Configuring Additional IP Addresses

Before you can assign IP addresses to VirtualMTAs, they must be recognized by the operating system. While normally one IP address is assigned to each NIC (Network Interface Card), it is also possible to define additional (*alias*) IP addresses to an existing NIC.

On Windows 2000, you add an alias IP address by opening the Control Panel, then right-clicking on "Local Area Connection", opening "Properties", double-clicking on "Internet Protocol", clicking "Advanced" and finally clicking "Add..." in the top half of the "IP Settings" tab. For more IPs, simply click on "Add..." multiple times, supplying the new IP addresses. The changes should normally become active as soon as you click on "OK" to close the Local Area Connection's `Properties` window.

On Red Hat Linux, you add an alias IP address by copying `/etc/sysconfig/network-scripts/ifcfg-eth0` (substituting the interface name accordingly if it is not `eth0`) to `ifcfg-eth0:1` (in the same directory) and then edit the file, changing `DEVICE` to `eth0:1` and `IPADDR` to the new IP. If the IP address you are adding is not on the same subnet as the primary IP address, you may need to update `NETMASK` as well. You can simply delete

the `NETWORK` and `BROADCAST` fields, if present, as they are automatically computed by the system. For more IPs, create additional files, named `ifcfg-eth0:2`, etc. To activate your changes, either stop and restart the interface or simply reboot the system.

Naturally, any IP addresses you add must be valid on the LAN to which your system is attached. Please consult with your network administrator if not sure what IP addresses to use.

8.6.1 Adjusting your Firewall Configuration

If PowerMTA is running on a machine behind a firewall, once it is configured to use diferent IP addresses for each VirtualMTA, your firewall will see outbound connection attempts from those IP addresses. You may need to adjust your firewall configuration accordingly.

8.7 DNS Prerequisites

Many ISPs check the host name and source IP addresses used in incoming SMTP connections against the information in the DNS to help determine legitimacy. Because of this, it is imperative that you have your DNS configured consistently with your usage of VirtualMTAs.

9. Mailmerge Support

9.1 Overview

PowerMTA includes internal mailmerge support to maximize performance and efficiency in the creation, queueing, and delivery of customized e-mail messages. PowerMTA's mailmerge implementation (PMTA-MM) supports both simple keyword or phrase merging as well as merging of larger blocks of data (such as paragraphs or body parts) for respective recipients. Merging does not depend on the actual message contents and can thus be used for both text and HTML formatted messages.

The process is basically similar to general mailmerge implementations. Your application generates a preformatted merge file or *recipe* that includes all of the recipient data (recipient addresses, names, order information, etc.) to be merged for each recipient, along with a message template. The message template includes the respective pointers to the variable content and data blocks to be merged in. Based on this, PowerMTA then creates the final merged (i.e., customized) messages for each recipient at actual delivery time.

PMTA-MM is supported with all of the current mail submission methods. Mailmerge submissions through SMTP and the pickup directory (BSMTP) are supported through special protocol extensions.

9.2 Benefits

While PowerMTA is designed to handle the queueing and delivery of very large volumes of customized messages, inline mailmerge support via PMTA-MM offers additional performance and efficiency advantages to all components of your solution. Some the obvious benefits are:

- decreased responsibility and load on your CRM or e-mail marketing application by shifting the merging to PowerMTA;
- increased feeding performance by application to PowerMTA, given that you would be feeding far fewer messages to PowerMTA for the same number of recipients;
- increased efficiency and delivery throughput given the decrease of the I/O impact on your PowerMTA machine. There will be a huge decrease in the number of queued mail files in the spool for the same number of recipients, allowing for maximum scalability. For example, instead of one spool file for each recipient in the queue, with PMTA-MM you could have 10,000 recipients in a single spool file.

9.3 Configuration Requirements

No special configuration is needed to submit mailmerge messages through the submission APIs or the pickup directory. However, to use the mailmerge SMTP extensions, you must enable them with the `allow-mailmerge` directive for the IP address(es) from which you will be submitting the messages:

```
<source 127.0.0.1>
    allow-mailmerge yes
</source>
```

9.4 Creating Mailmerge Messages

9.4.1 Message Template and Mailmerge Variables

Creating mailmerge messages is relatively easy. The first task is to create a message template, marking and labeling the portions that you would like customized for each recipient, like in the example below:

```
(other headers omitted)
From: shipping-confirmation@example.port25.com
Subject: Your order #[OrderNumber] shipped!

[First_Name],

Your order #[OrderNumber] was shipped on [ShipTime].  Thank
you for your
business!
```

The labels for the customized portions are names of the `variables` which define what the content for each portion.

Once you have created your message template, you define the content for of these variables for the various recipients. For example, for one recipient you may want to have `First_Name` set to `Bob`, `OrderNumber` set to `1234-56` and `ShipTime` to `7/14 10:00`. In the example above, this would yield:

```
(other headers omitted)
From: shipping-confirmation@example.port25.com
Subject: Your order #1234-56 shipped!

Bob,

Your order #1234-56 was shipped on 7/14 10:00.  Thank you for
your
business!
```

Mailmerge variables are defined per recipient, and you cannot use the same variable name to define more than one set of data in the same message. Variable names are case-insensitive, e.g. `name` and `Name` are the same variable. Variable names may be composed of the letters (a-z and A-Z), the digits (0-9), and the underscore (_). The first character in a variable name must be a letter.

As noted above, variable substitutions are enclosed in square brackets in the message template. To include a square bracket itself, simply write it twice. For example,

```
[First_Name],

Please fill out the fields below marked with square brackets
[[].
...
```

would become (for a First_Name of "George")

```
George,

Please fill out the fields below marked with square brackets
[]
...
```

9.4.2 Reserved Variables

Variable names beginning with an asterisk are "reserved" and have a special meaning to PowerMTA:

*from
> overrides the SMTP MAIL FROM (originator) address for the recipient's message;

*envid
> overrides the DSN envelope ID for the recipient's message, for better tracking in bounces and in the accounting file. This setting overrides any other method of defining the envelope ID;

*vmta
> selects the VirtualMTA (or VirtualMTA pool) through which the recipient's message should be delivered. This setting overrides any of the other VirtualMTA selection methods described in Section 8.4;

*jobid
> defines the job (or campaign) ID for the recipient's message, for tracking in the WWW based monitoring facility. If set, the content of this variable overrides any job ID already set for the recipient's message. x-job header can be used with Mailmerge, but using *jobid is faster.

*parts
> defines the list of merge parts to include in the recipient's message. For example, if your message has 3 parts, numbered 1, 2 and 3 and you want to include all of

237

them in the recipient message, you should set `*parts=1,2,3` or shorter, `*parts=1-3`. If you wanted part 3 to go before part 2 in the recipient's message, you would set `*parts=1,3,2`;

*to

merges in the defined RCPT TO/recipient e-mail address. This variable is automatically set from the recipient address and cannot be overridden;

*date

merges in the message's reception date & time stamp. This variable is automatically set and cannot be overridden;

*schedule

sets the delivery schedule for the recipient in a message. This overrides any schedule that may be defined in the message body.

9.4.3 Merge Parts

Each message template is made up of one or more sections called *merge parts*. A merge part can be a sentence, a paragraph, a block of paragraphs, a body part, or even the message headers. Within each merge part there can be many mailmerge variable substitutions.

The parts or sections of the message are labelled with a part number, and then assembled for each recipient based upon the content of the special `*parts` variable for the recipient.

For example, a travel alerts service could send messages in three parts: the first containing the headers and a common preamble,

```
From: "Customer Service" [*from]
To: "[FName]" [*to]
Date: [*date]
Subject: Travel Plans

Dear [FName],

You are flying out of [Airport] given that it is closest to
your location
in [City], [State].
```

the second containing a notice that is only sent to some of the recipients,

```
With this flight, you are only [MilesToBonus] frequent flyer
miles away from
your next SuperBonus(TM)!
```

the third with a common final block:

```
Have a great trip!
```

238

Some recipients could then be sent all parts (`*parts=1-3`) while others would only be sent parts #1 and #3 (`*parts=1,3`). Note that the variable `MilesToBonus` would only need to be defined for those recipients receiving part #2.

9.5 Submitting Mailmerge Messages

You can feed mailmerge messages to PowerMTA using any the standard supported submission APIs, pickup directory, or through PowerMTA's mailmerge SMTP extensions. It is recommend to add no more than 5000 recipients per mail merge file when feeing via pickup directory.

9.5.1 API Submission

All PowerMTA submission APIs include support for mailmerge. The basic process involves defining per-recipient variables by invoking the `defineVariable` for each `Recipient` object, and then passing the message template by invoking the `addMergeData` method for the `Message` object. Please refer to the corresponding API reference for more details.

Sample programs for using mailmerge in the APIs are provided with PowerMTA.

9.5.2 Mailmerge SMTP extensions

Port25 has extended the SMTP protocol for sites that would prefer to feed PowerMTA mailmerge messages via a modified SMTP interface.

The mailmerge extensions define three new verbs:

XMRG

> Replaces the `MAIL` command of a standard SMTP submission. All extensions (such as DSN, VERP, etc.) that are supported with `MAIL` are also supported with `XMRG`.
>
> Example:
>
> ```
> XMRG FROM:<travelAlerts@example.port25.com> VERP
> ```

XDFN

> Defines variables for the recipient following it. One or more `XDFN` commands should be issued prior to each `RCPT` command in a mailmerge submission.
>
> Variables are defined in the format *name=value*, where *name* is the variable name and *value* the value to to be associated with the variable. If the value contains any white space, it should be enclosed in double quotes. You can specify multiple variables in a single `XDFN` command by separating them with blanks.

239

Example:

```
XDFN name="Bob Example" Airport=BWI
XDFN *jobid="TravelAlerts"
RCPT TO:<bob@example.port25.com>
```

XPRT

Replaces the DATA command of a standard SMTP submission. One XPRT command should be issued per merge part in the mailmerge submission, passing the merge part number, like in the example below:

```
XPRT 1
```

Like for DATA, PowerMTA responds to the XPRT command with a 3xx reply code. Also, like data, the message template must be passed dot-escaped. A dot on a line by itself terminates the merge part.

The last XPRT is indicated with the additional LAST parameter and terminates the submission:

```
XPRT 2 LAST
```

The following example shows all the three verbs used to submit a mailmerge to two recipients. In addition to parts 1 and 3, Alice is also sent part #2 with a reminder that she's close to obtaining a bonus:

```
XMRG FROM:<travelAlerts@example.port25.com> VERP
XDFN FName="Alice" Airport="IAD" MilesToBonus="120"
XDFN City="Herndon" State="VA" *parts=1-3 *vmta="mta4"
XDFN *jobid="TravelAlerts"
RCPT TO:<alice@example.port25.com>
XDFN FName="Bob" Airport="BWI"
XDFN City="Ellicott City" State="MD" *parts=1,3 *vmta="mta5"
XDFN *jobid="TravelAlerts"
RCPT TO:<bob@example.port25.com>
XPRT 1
From: "Customer Service" [*from]
To: "[FName]" [*to]
Date: [*date]
Subject: Travel Plans

Dear [FName],

You are flying out of [Airport] given that it is closest to
your location
in [City], [State].
.
XPRT 2

With this flight, you are only [MilesToBonus] frequent flyer
miles away from
```

240

```
your next SuperBonus(TM)!
.
XPRT 3 LAST

Have a great trip!
.
```

9.5.3 Pickup Directory Submission

**Do not use the pickup directory submission method if you expect high performance
submission.

The pickup directory submission method includes support for mailmerge, which greatly
minimizes the disk I/O impact usually associated with this submission method. Instead of
creating and writing 100,000 messages, one per recipient, mailmerge support allows you
to write say only 25-50 files for the same number of recipients, greatly minimizing the
resources utilized and increasing throughput. Section 7.4 covers the pickup directory
more in detail.

Support for mailmerge in the pickup directory is accomplished through its support for
BSMTP (batched SMTP) files. See Section 7.4.1 for more information.

Here is a simple example of a properly formatted BSMTP mailmerge file for
PowerMTA's pickup directory:

```
XMRG FROM:<bounce@port25.com>
XDFN *from="blahblah@port25.com"
XDFN subject="BSMTP mailmerge format"
RCPT TO:<bob@port25.com>
XPRT 1 LAST
From:     [*from]
To:       [*to]
Subject:  [subject]

Hello [*to]!
.
```

Here is another that contains information for 4 recipients:

```
XACK ON
XMRG FROM:<bounce@port25.com> verp
XDFN FName="Jim" FavFood="pizza" Car="minivan" CID="1234"
XDFN *vmta="mta1" *parts="1-3" *jobID="1234"
RCPT TO:<jim@test.port25.com>
XDFN FName="Tim" FavFood="sushi" Car="porsche" CID="2345"
XDFN *vmta="mta2" *parts="1,3,2" *jobID="5678"
RCPT TO:<tim@test.port25.com>
XDFN FName="Cosmo" Car="Gremlin" CID="3456"
```

```
XDFN *vmta="mta3" *parts="1,3" *jobID="1234"
RCPT TO:<cosmo@test.port25.com>
XDFN FName="Frank" FavFood="pizza" CID="4567"
XDFN *vmta="mta5" *parts="1,2" *jobID="5678"
RCPT TO:<frank@test.port25.com>
XPRT 1
X-CID: [CID]-abc
From: "Customer Service" [*from]
To: "[Fname]" [*to]
Subject: Travel Plans
Date: [*date]

Hey [FName],
.
XPRT 2
Your favorite food is [favfood].
.
XPRT 3 LAST
You drive a [car].
.
```

Section 9.5.2 also provides another good example for reference.

Here is an example of a multi-part mailmerge file for a pickup directory:

```
XMRG FROM:<me@yourdomain.com> verp
XDFN *from="me@yourdomain.com"
XDFN subject="test  \ 1"
RCPT TO:<you@yourdomain.com>
XDFN *from="me@yourdomain.com" subject="test 2"
RCPT TO:<you@yourdomain.com>
XPRT 1 LAST
Mime-Version: 1.0
Content-Type: multipart/alternative; boundary="12345"
From:      [*from]
To:        [*to]
Subject:   [subject]

--12345
Content-Type: text/plain;

Hello.  This is the text version.
--12345
Content-Type: text/html;

<b>This is the HTML version.</b>

--12345--
.
```

242

9.5.4 The XACK SMTP extension

To facilitate and improve the feeding performance while using the mailmerge extensions, Port25 has also defined the XACK protocol extension.

The XACK command disables acknowledgements for the RCPT and XDFN commands. It accepts a single, parameter, either ON or OFF. To disable acknowledgements, you issue XACK like in the example below:

```
XACK OFF
```

Once acknowledgements are disabled, they stay disabled until the connection is closed or until they are re-enabled with XACK ON.

While acknowledgements are disabled, any errors occurring as a result of RCPT or XDFN command during a submission (i.e., after MAIL or XMRG and before the final DATA/BDAT/XPRT command) are reported as a response to the next DATA or BDAT (for non-mailmerge messages) or XPRT (for mailmerge messages). As a result of the error, the entire submission is cancelled.

Some errors (e.g., address errors in the RCPT) may optionally be reported by sending a failure Delivery Status Notification, thus allowing the submission to continue. Any RCPT or XDFN errors (e.g., out of sequence errors) occurring not during a submission are ignored entirely.

The following example shows the connection trace (with commands and replies, but with the message content omitted) while submitting a mailmerge message:

```
>>> 220 mailhost.port25.com (PowerMTA v2.x) ESMTP service ready
<<< EHLO test.port25.com
>>> 250-mailhost.port25.com says hello
>>> 250-ENHANCEDSTATUSCODES
>>> 250-PIPELINING
>>> 250-CHUNKING
>>> 250-8BITMIME
>>> 250-XACK
>>> 250-XMRG
>>> 250-SIZE 0
>>> 250-VERP
>>> 250 DSN
<<< XACK OFF
>>> 250 2.0.0 ok
<<< XMRG FROM:<bounce@example.port25.com> verp
>>> 250 2.1.0 XMRG ok
<<< XDFN FName="Jim" FavFood="pizza" Car="minivan" CID="1234" *vmta="vmtapool" *parts="1-3"
<<< RCPT TO:<jim@example.port25.com>
<<< XDFN FName="Tim" FavFood="sushi" Car="porsche" CID="2345" *vmta="vmta2" *parts="1,3,2"
<<< RCPT TO:<tim@example.port25.com>
<<< XDFN FName="Cosmo" Car="Gremlin" CID="3456" *vmta="vmta3" *parts="1,3"
<<< RCPT TO:<cosmo@example.port25.com>
<<< XDFN FName="Frank" FavFood="pizza" CID="4567" *vmta="vmta1" *parts="1,2"
<<< RCPT TO:<frank@example.port25.com>
<<< XPRT 1
>>> 354 send part
<<< .
>>> 250 2.6.0 part ok
<<< XPRT 2
>>> 354 send part
<<< .
>>> 250 2.6.0 part ok
<<< XPRT 3 LAST
>>> 354 send part
<<< .
>>> 250 2.6.0 message received
<<< QUIT
>>> 221 2.0.0 mailhost.port25.com says goodbye
```

10. Advanced Features

10.1 Sender Reputation Monitoring

10.1.1 Overview

One of the keys to maximizing deliverability to valid domains is to know as soon as possible when specific delivery problems are occurring. The sooner one realizes that there are certain problems, or more specifically, that a remote site is rejecting your messages for one reason or another, the sooner one can attempt to get the problems resolved. There are many different kinds of errors that remote sites can respond with when connected, both positive and negative, and these responses allow you to see first hand your current sender reputation, in the eyes of that recipient domain or mail gateway.

Given the importance of having this information as soon as possible, PowerMTA v3.0 includes automated sender reputation monitoring, via an advanced, real time SMTP command monitoring implementation. It not only gives you an early warning alert system for specific types of errors, but it can be configured to try and possibly resolve some problems as well, via support for automated configuraton changes for future delivery attempts.

10.1.2 Implementation

For greatest flexibility and control, PowerMTA groups messages for a particular domain for a particular VirtualMTA in its own queue. Sender reputation monitoring provides real time monitoring of defined SMTP responses (e.g., "551 Sender IP rejected") from remote hosts for each queue, with the ability to define certain actions (e.g., send an e-mail alert while throttling deliveries) upon a match.

This is accomplished through the introduction of queue "modes". The two modes initially supported are "normal" and "backoff". You can configure PowerMTA so that, when a queue enters backoff mode, it

- sends an email alert to one or more addresses (`backoff-notify` directive);
- dynamically changes the retry schedule for that queue (`backoff-retry-after` directive);
- dynamically applies per-queue delivery throttling (`backoff-max-msg-per-hour` directive);
- dynamically reroutes the messages from one VirtualMTA to another VirtualMTA (`backoff-reroute-to-vmta` directive).

The dynamic reroute, combined with the ability to now have separate per domain configurations per VirtualMTA, allows one to use the full range of domain directives beyond the new `backoff-...` directives (e.g., using a new `bounce-after`), as well as VirtualMTA attributes, for the fallback queue. A chain of backoff VirtualMTAs is also supported (vmta -> vmta1, vmta1 -> vmta2, vmta2 -> vmta3, etc.).

SMTP response matching is performed based on a list of perl-compatible regular expressions, entered in a `smtp-pattern-list` directive in the configuration file.

For example, you could define a list of errors that might occur during deliveries to AOL in a `smtp-pattern-list` named `aol-errors`, and specify that the queue be put in backoff mode if any such match:

```
<smtp-pattern-list aol-errors>
    reply /generating high volumes of.* complaints from AOL/    mode=backoff
    reply /Excessive unknown recipients - possible Open Relay/  mode=backoff
    reply /^421 .* too many errors/                             mode=backoff
    reply /blocked.*spamhaus/                                   mode=backoff
    reply /451 Rejected/                                        mode=backoff
</smtp-pattern-list>
```

You would then indicate that PowerMTA should match based on that pattern list while delivering to `aol.com`:

```
<domain aol.com>
    ...
    smtp-pattern-list aol-errors
    ...
</domain>
```

Finally, also within that `domain` definition, you would specify certain `backoff-...` actions to be taken when the queue enters backoff mode:

```
<domain aol.com>
    ...
    smtp-pattern-list        aol-errors  # SMTP response pattern list to use
    backoff-max-msg-per-hour 120         # 10 messages per 5 min interval
    backoff-retry-after      1h          # retry every hour
    backoff-notify           postmaster@yourdomain.com
    ...
</domain>
```

Upon the first match of the SMTP response, the queue for `aol.com` for this particular VirtualMTA will enter backoff mode, dynamically applying the specific `backoff-...` directives:

- send an email alert to the defined PowerMTA postmaster address;
- change the retry interval to `aol.com` for this particular VirtualMTA to 1 hour;
- throttle the delivery rates to `aol.com` for this particular VirtualMTA to at most 120 attempted deliveries per hour;

The backoff directives apply until the mode for the queue is returned to normal, which can be accomplished either within a `smtp-pattern-list` (by specifying a `mode=normal` for a pattern), with the `pmta set queue` command, or by restarting PowerMTA. In the example above, the command to return that queue to normal mode would be

```
pmta set queue --mode=normal aol.com
```

Defining backoff VirtualMTAs is also relatively straightforward, using the `backoff-reroute-to-vmta` directive. It simply specifies that PowerMTA should reroute messages to the given VirtualMTA in case the queue enters backoff mode.

For example, if messages are queued for `example.port25.com/vmta1` and that queue enters backoff mode while `backoff-reroute-to-virtual-mta` is set to `vmta2`, all of its messages would be rerouted to the queue `example.port25.com/vmta2`. Since the messages would then be queued within `vmta2`, that VirtualMTA's settings would apply. New messages entering the queue are also immediately rerouted to the new VirtualMTA, as long as the `vmta1` queue remains in backoff mode.

For example, one could define a backoff VirtualMTA as follows:

```
<virtual-mta mta1>
    smtp-source-host 1.1.1.1 mta1.port25.com

    <domain *>
        backoff-reroute-to-vmta  backoff-mta1
        smtp-pattern-list        block-errors
    </domain>
</virtual-mta>

<virtual-mta backoff-mta1>
    smtp-source-host 1.1.1.2 mta2.port25.com

    <domain *>
        retry-after   30m
        bounce-after  1h
    </domain>
</virtual-mta>
```

With this, if there is an SMTP pattern match for any domain for VirtualMTA `mta1`, any messages in that specific queue will be delivered using the domain directives and VirtualMTA parameters defined in VirtualMTA `backoff-mta1`. As desired, the backoff parameters will **only** apply to messages for the specific queue, that is, for messages in the queue whose delivery attempts matched a SMTP response.

Note that since the messages are really transferred from one VirtualMTA to another, the `backoff-reroute-to-vmta` directive effectively takes precedence over `backoff-max-msg-per-hour` and `backoff-retry-after`.

247

10.1.3 Supporting commands

Two new command line commands have been added to facilitate working with sender reputation monitoring and queue modes:

- `pmta set queue --mode={normal|backoff} domain/vmta`
- `pmta show queues`

The `set queue` command sets the given queue's mode of operation to either normal or backoff. See [Section 6.3.11](#) for more information.

The `show queues` command allows you to view current information on the specified queues, including the current mode of the queue. See [Section 6.3.13](#) for more information.

10.2 SMTP AUTH support

10.2.1 Overview

To facilitate creating a secure email delivery channel, PowerMTA supports authentication in both inbound and outbound SMTP connections.

10.2.2 Implementation for Inbound Connections

PowerMTA supports authentication with either the `PLAIN` or the `CRAM-MD5` mechanism in inbound connections, with the `DIGEST-MD5` mechanism being considered for a future version. Support for `CRAM-MD5` is available whenever `AUTH` itself is enabled. Since PowerMTA does not yet support encryption of connections and using the `PLAIN` mechanism would involve sending passwords unencrypted over the Internet, support for it must be explicity enabled with the `allow-unencrypted-plain-auth` directive. The password is never sent when using CRAM MD, as it's a challenge/response authentication.

To use `AUTH` you must also create `<smtp-user>` tag for each valid username, including their passwords, as well as the named source to use when that user authenticates, like in the example below:

```
<smtp-user jsmith>
  password qwerty123
  source auth2
</smtp-user>

<smtp-user jdoe>
  password abc123
  source auth1
</smtp-user>
```

For maximum flexibility in supporting the various SASL authentication methods, the passwords must be entered in plaintext. Because of this, it is crucial that the file permissions are set up as restrictively as possible. Port25 intends to look into ways to encrypt the passwords in a future release.

The "named" source in the configuration file (that is a `<source>` entry that has a name rather than an IP address or CIDR specification), allows you to specify which `<source>` settings should apply on a per-user basis.

In the example above, and with the configuration file below, user `jsmith` would be allowed access to PowerMTA's mail merge extensions, while `jdoe` would not:

```
<source auth2>
  allow-mailmerge yes
  always-allow-relaying yes   # allow feeding for auth2
  process-x-virtual-mta yes   # allow selection of a VirtualMTA
  max-message-size 0          # 0 implies no cap, in bytes
  smtp-service yes            # allow SMTP service
  require-auth true
</source>

<source auth1>
  allow-mailmerge no
  always-allow-relaying yes   # allow feeding for auth1
  process-x-virtual-mta yes   # allow selection of a VirtualMTA
  max-message-size 0          # 0 implies no cap, in bytes
  smtp-service yes            # allow SMTP service
  require-auth true
</source>

<source 0/0>
    allow-unencrypted-plain-auth yes
</source>
```

When users `jsmith` and `jdoe` authenticate, any settings in their named sources would override those settings normally obtained based on each connections' source IP address. This way, you can combine the usual IP-based settings (which apply to all connections) with specific settings based on the authenticated user.

See [Section 3.2.4](#) for more details on SMTP service directives.

10.2.3 Implementation for Outbound Connections

PowerMTA supports authentication with the `CRAM-MD5`, `LOGIN`, `or PLAIN` mechanisms in outbound connections, with additional mechanisms being considered for a future version.

To have PowerMTA authenticate to a remote mailer when delivering messages to a domain, define the domain in the configuration file and specify the `auth-username` and `auth-password` directives. Authentication will be attempted only if the remote mailer supports `CRAM-MD5`, `LOGIN`, `or PLAIN` (`LOGIN or PLAIN if using the <domain>` directive "use-unencrypted-plain-auth yes").

To illustrate how this feature could be used, consider the following scenario: an email service provider using PowerMTA is delivering email on behalf of, and thus "from" `secure.example.org`, but that domain's corporate mail system will only accept mail using that domain as the originator from an authenticated source.

To satisfy the corporate mail system's requirement, one could configure PowerMTA to select a VirtualMTA based on pattern matching of the `From:` address (using the support for pattern matching of any RFC 2822 header), and within that VirtualMTA, configure the required username and password for all mail delivered to `secure.example.org`.

The VirtualMTA definition could look like:

```
<virtual-mta auth-mta>
    smtp-source-host 1.2.3.4 vmta.esp.com
    ...

    <domain secure.example.org>
        auth-username costanza
        auth-password bosco
    </domain>
</virtual-mta>
```

With this, all mail delivered by this VirtualMTA to the domain of `secure.example.org` would attempt authentication, whereas all other mail delivered to this domain by other VirtualMTAs would not.

Note that authentication is not attempted if `auth-username` and `auth-password` are not defined for a domain.

10.3 Sender-ID support

10.3.1 Overview

PowerMTA v3.0 includes automatic support for the Sender-ID Framework backed by Microsoft, as it relates to the sending of authenticated e-mail. Sender-ID is a specification created to counter e-mail domain spoofing and to provide greater protection against phishing schemes. It helps verify that each e-mail message comes from a particular domain from which it claims to come, based on the IP address of the sending server. More information about Sender-ID can be found at http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx.

10.3.2 Automated Authentic

To help maintain a strong sender reputation, PowerMTA can be configured to force Sender-ID compliance, by only delivering mail that is Sender-ID compliant.

PowerMTA v3.0 includes the ability to automatically check each mail submitted to make sure that it is Sender-ID compliant before delivery is attempted. Compliance checking is done based on the sender address, the source IP address to be used in the connection, and the SPF 2.0 records published in the DNS. The sender address used depends on the scope of the test being made: both the "pra" (Purported Responsible Address, based on message headers) and the "mfrom" (based on the SMTP `MAIL FROM` and `HELO` domains) scopes are supported. Other compliance checks, such as for DomainKeys are being considered.

If the test results in anything but a "Pass" or a "TempError", the message bounces, with an appropriate bounce sent to the `MAIL FROM` address.

To enable Automated Authentic for a domain, set the `check-mfrom-outbound` directive to `yes` in the domain's defintion in the configuration file (or in `<domain *>` for all domains).

When enabling this feature, it is advisable to disable it for the domains to which your bounces are delivered, if they are delivered via SMTP. This way, if the Sender-ID check fails, there is no risk of these bounces being lost ("double-bouncing") due to some problem in your bounce domain's SPF records.

For example, if your `MAIL FROM` domain was `yourbouncedomain.com`, you could configure:

```
<domain yourbouncedomain.com>
    check-mfrom-outbound no
</domain>

<domain *>
    check-mfrom-outbound yes
</domain>
```

251

Note that since a specific IP address is needed for the Automated Authentic check, you will need to make sure that the IP address(es) used by PowerMTA for outbound connections are defined in the configuration file (via the `smtp-source-host` directive). If no specific source IPs are defined, PowerMTA cannot perform the Sender-ID check. It then generates an error message in the log file and treats this like a transient ("4xx") error, keeping the messages in the queue and retrying until the `bounce-after` interval passes.

10.3.3 SUBMITTER Optimization

PowerMTA v3.0 senses when the new SUBMITTER extension is supported by the receiving mailer and automatically computes the message's PRA and passes it on, so that the authentication check can be performed before the message content is transferred.

Since both detection and PRA computation are performed automatically, no configuration is necessary to activate this feature. It is always active when applicable.

10.4 DomainKeys/DKIM support

10.4.1 Overview

Port25 Solutions is one of the first companies to support an early implementation of "DomainKeys" and the later revision, DKIM, the e-mail sender authentication scheme backed by Yahoo! based on cryptographically signing e-mail messages. Port25 highly recommends that all users looking to use this functionality read through the DomainKeys & DKIM specification currently found here:

http://domainkeys.sourceforge.net/ & http://www.dkim.org/.

Here is a brief description of how an email is signed with DomainKeys & DKIM:

1. A public/private key pair is generated for use for signing all outgoing messages;
2. The public key is published in DNS, and the private key is made available to the outbound email servers;
3. When a message is submitted for delivery, the email system uses the stored private key to generate a digital signature of the message before delivery and adds it to the message in the `DomainKey-Signature:` header.

10.4.2 PowerMTA's DomainKeys/DKIM Implementation

PowerMTA supports DomainKeys & DKIM signing of email on a per VirtualMTA or a global basis. That is, within each `<virtual-mta>` definition you can specify a domain

key to use. Alternatively, domain keys can be configured at the global level and they will be used by any VirtualMTA's. The new directive is named `domain-key` and takes three arguments: the DomainKeys selector, the domain (or parent domain) from which messages will originate, and the name of the file containing the private key. For example:

```
<virtual-mta domainkey>
    host-name port25.com
    ...
    domain-key test,port25.com,c:\pmta\test.port25.pem
</virtual-mta>
```

or

```
domain-key test,port25.com,c:\pmta\test.port25.pem
```

You can specify multiple `domain-key` entries. PowerMTA determines the key to use by going through the keys sequentially and picking the first key whose domain either equals or is a parent of the message's *sending domain*. If there are keys that match at both the global level and the VirtualMTA level, the `<virtual-mta>` domain keys will be used. The sending domain is that of the `Sender` header, or, if `Sender` is not present, of the `From` header. If no keys match the sending domain, the message is not signed.

For DKIM, if the sender or from headers are not able to be set or changed (e.g. An ESP customer's email that cannot be changed), use of the dkim-identity <domain> directive allows for specifying a domain or email address to be used for the signing and authentication in place of the sender or from header. Using dkim-identity sets the i= value in the DKIM signature to the value specified in with dkim-identity, and the d= value to that set by the domain-key directive. Use of this directive will not affect message signed with DomainKeys.

To set up DomainKeys/DKIM signing for a domain,

1. if this is your first domain key, define the `_domainkey` TXT record in your DNS, specifying your DomainKeys policy:

   ```
   _domainkey.example.org. 1D IN TXT "t=y\; o=~\;"
   ```

2. Please do not use the example above unchecked; refer to the DomainKeys specification for more information on the policy record syntax.
3. Determine the domain in which the key will be registered. That should be either the domain itself or a parent of the sending domain. In the examples below, we will assume `example.org`.
4. Decide what to use for your selector. The selector will uniquely identify the key within the domain determined above. We will use `k1` in the examples below.

5. Generate your private key using the port25 DomainKey wizard:
   at https://www.port25.com/support/domainkeysdkim-wizard/
   (If using the wizard, skip step 8)
   or use `openssl`:

```
openssl genrsa -out k1.example.pem 1024
```

6. This creates a 1024-bit private key in the file `k1.example.pem`. You can use any other file name, as long as you modify the following steps accordingly;
7. Move that file to `c:\pmta\` on Windows or to `/etc/pmta` on Unix and ensure its permissions are set properly (ideally only PowerMTA should have access to it).
8. Generate a public key from your private key:

```
openssl rsa -in k1.example.pem -out public.pem -pubout
```

9. Define a TXT record in the DNS for the domain key, using the public key generated in the last step in the `p` tag:

```
K1._domainkey.example.org 1D IN TXT "k=rsa\; p=MIGfMA0...\;"
```

10. Please refer to the DomainKeys specification for more information on the record syntax.
11. Add the `domain-key` directive to the config file:

```
<virtual-mta example>
    ...
    domain-key k1,example.org,c:\pmta\k1.example.pem
</virtual-mta>
```

12. or

```
domain-key k1,example.org,c:\pmta\k1.example.pem
```

or

```
domain-key k1,*,c:\pmta\k1.example.pem
```

13. Enable signing for the domain(s) desired, or in `<domain *>` to enable signing for all domains:

```
<domain yahoo.com>
    ...
    dkim-sign yes
```

```
</domain>
```

14. If the `domain-key` directive is configure for a VirtualMTA instead of globally, configure PowerMTA so that email originating in the domain(s) to be signed select that VirtualMTA, for example, by using a pattern list:

```
<pattern-list L>
    mail-from /[@\.]example\.com$/   virtual-mta=example
</pattern-list>

<source 10.1.0.0/16>   # feeding sources
    pattern-list L
</source>
```

15. See Chapter 8 for more information on selecting VirtualMTAs. Section 3.2.16 describes pattern lists in greater detail.

10.4.3 DKIM signing based on values in a header

PowerMTA supports the ability to sign messages with DKIM using values passed in the header **x-dkim-options**. This use of this directive requires the use of the <source> directive "**process-x-dkim-options true**".This header may contain the selector, domain, or identity used for DKIM signing:

```
x-dkim-options: <semicolon separated list of options>
```

Options:

- **s** - selector- can be any value other than empty string
- **d** - domain - should be a domain
- **i** - identitiy - can be a complete email address or an email address minus the local part (e.g. @example.com).

To use this header, one must specify either **i** or **d**. If both are specified, the value of **d** will be used to select amongst the keys, and not **i**. In addition, the value of **s** can be specified as the selector. If **i** or **d** is specified, it will be treated as the identity to look up the key. If neither is specified, PMTA will use the current method of key selection (sender-or-from). If **s** is specified, only the key with the matching selector among the list of keys matching the identity will be used to sign. If no such key for the selector is found, the email is not signed. The order in which options are specified is not important.

For Example, with the following configuration:

```
domain-key key1,example.com,c:\pmta\key1.example.com.pem
```

The **x-dkim-options** header will have to specify s=key1 and d=example.com (or specify an **i** with the same domain, or a subdomain):

```
x-dkim-options: s=key1;d=example.com
or
x-dkim-options: s=key1;i=example.com
```

It is important to note that if a value for **d** is specified, then only the key with the EXACT domain or the first key which matches every domain ('*') will be used. Keys for subdomains are not used for signing such emails. Also, note that **x-dkim-options** overrides **dkim-identity**, so **dkim-identity** is not used if either **d** or **i** values are specified in the **x-dkim-options** header.

Use of the x-dkim-options header is an all or nothing usage. If it matches, it is used, if it doesn't match, no signing is done.

Lastly, **dkim-identity-fallback** is used as the fallback key to sign with if no matching key has been found, irrespective of whether **x-dkim-options** or **dkim-identity** was used to specify the identity to sign with.

10.4.4 Second DKIM signing support

PowerMTA now officially supports double DKIM signing a message. This is a direct requirement of gmail when using their feedback loop. All of the existing dkim-* directives now support the prefix "second", and which are used for the second signature (bottom most one) for the message.

We use the same domain-key directives for both signatures. Here is the list of new directives used for the second DKIM signature:

```
second-dkim-add-body-limit
second-dkim-add-timestamp
second-dkim-algorithm
second-dkim-body-canon
second-dkim-expire-after
second-dkim-headers
second-dkim-headers-canon
second-dkim-identity
second-dkim-sign
second-dkim-query-method
```

These new directives are used just like their equivalent dkim-* directives.

A configuration example is below which sends messages with both a customer signature and ESP signature, which are only to be applied to messages in VirtualMTA 12345. The customer signature is based on the domain of the From: header (customerdomain.com) while the ESP signature is based on the second-dkim-identity option defined (espdomain.com). PowerMTA would include gmail's required Feedback-ID header in both signatures for messages to gmail only.

```
<virtual-mta 12345>
    domain-key esp,espdomain.com,c:\pmta\keys\esp.espdomain.com.pem
    domain-key
customer,customerdomain.com,c:\pmta\keys\customer.customerdomain.com.pem
    <domain gmail.com>
       dkim-headers Feedback-ID
       second-dkim-headers Feedback-ID
    </domain>
    <domain *>
       dkim-sign  yes
       second-dkim-sign  yes
       second-dkim-identity @espdomain.com
    </domain>
</virtual-mta>
```

Note: The x-dkim-options header method is not supported for the second signature, but still can be used for the first signature.

PowerMTA will still deliver the message if due to message attributes and configuration settings, there are no signatures to be applied, only one signature to be applied or both signatures are to be applied.

10.5 Warming of IPs with Cold VirtualMTA

10.5.1 Overview

**This functionality works when you have at least one warm IP address. If you only have cold IP addresses then it makes no sense to use this specific functionality in PowerMTA. Instead, if you have nothing but cold IPs, use rate limiting in the VirtualMTAs .

When starting to send email from a new IP address, it is sometimes important to control the amount of traffic sent from that IP, so that reputation can be built properly. To facilitate this task, PowerMTA adds support for configuring separate queues and

diverting part of the traffic to those queues.

The cold VirtualMTA functionality, at its core, is a rerouting feature. **The first x messages submitted to the warm VirtualMTA each day are auto-rerouted to the cold VirtualMTA.** The number that is rerouted for each domain per day is determined by the "max-cold-virtual-mta-msg" directive defined in the warm VirtualMTA for the domains. The rest of the messages submitted that day to the warm VirtualMTA stay in the warm VirtualMTA. The cold VirtualMTA rerouting counters are reset at midnight.

The messages routed to the cold VirtualMTA are handled/delivered based on the directives defined in the cold VirtualMTA. To this end it is important to configure the cold VirtualMTA as desired.

For example:

```
<virtual-mta vmta1>
  smtp-source-host 1.2.3.4 vmta1.example.com
  cold-virtual-mta vmta2
    <domain *>
       max-cold-virtual-mta-msg 1000/day
    </domain>
</virtual-mta>

<virtual-mta vmta2>
    smtp-source-host 5.6.7.8 vmta2.example.com
    <domain *>
       max-msg-rate  100/h
    </domain>
</virtual-mta>
```

With these settings, the first 1000 messages submitted into the warm VirtualMTA `vmta1` per day, for each domain, would be auto-rerouted to the cold VirtualMTA `vmta2`. Every message submitted after the first 1000 per day per domain to the warm VirtualMTA would stay in the warm VirtualMTA, with the daily rerouting counters resetting at midnight. Because **max-cold-virtual-mta-msg setting is per domain**, the first 1000 submitted for yahoo.com to `vmta1` per day would be rerouted to `vmta2`, the first 1000 submitted for hotmail.com to `vmta1` per day would be rerouted to `vmta2,` etc.

This is the count for each domain separately given that "max-cold-virtual-mta-msg 1000/day" is defined in <domain *> in the VirtualMTA in this example. If different counts per day for different domains are needed, define the different domain definitions in the VirtualMTA with the specific counts desired for each, along with a count for all other domains in <domain *>. Messages that are routed to the cold VirtualMTA are then handled based on the settings defined in the cold VirtualMTA.

The default setting for the "max-cold-virtual-mta-msg" directive is 0, which means "don't re-select a cold VirtualMTA". If "max-cold-virtual-mta-msg" is set globally but the

258

VirtualMTA selected doesn't have a "cold-virtual-mta" directive configured, no re-selection takes place.

Since the cold VirtualMTA is just another VirtualMTA, this feature can also be used to route traffic from an old server to a new server. This is useful when moving server locations and the new server is getting new IPs that are not yet established with a good reputation. For example:

```
<virtual-mta vmta1>
  smtp-source-host 1.2.3.4 vmta1.example.com
  cold-virtual-mta vmta2
  <domain *>
    max-cold-virtual-mta-msg 1000/day
  </domain>
</virtual-mta>

<virtual-mta vmta2>
  smtp-source-host 5.6.7.8 vmta2.example.com
  <domain *>
    queue-to route.queue
  </domain>
  <domain route.queue>
    route [9.8.7.6]:25
    max-smtp-out 20
  </domain>
</virtual-mta>
```

In the above example, the first 1000 messages per domain per day injected into vmta1 are routed to vmta2. At this point that are grouped into one large queue with the queue-to directive to help better manage the amount of outbound connections the messages will make. Without this setting 1000 domains may try to make 1000 connections to the new PowerMTA server.

Both the limits and the selection are persisted on disk, so that you can safely re-start PowerMTA without resetting the current rates.

10.6 Using of Bounce Category Patterns

10.6.1 Overview

This directive makes the bounce categories used in CSV accounting files and the X-PowerMTA-BounceCategory DSN field customizable. A customer can specify the tag in the configuration file, in which case the patterns specified there take precedence over those defined in config-defaults file. The data in the <bounce-category-patterns> tag consist of two fields. The first field is the SMTP response on which to match, and the

second field is the name of the category to be used. Custom category names are allowed.
It should be entered in a <bounce-category-patterns> entry, like in:

```
<bounce-category-patterns>
    /spam/ spam-related
    /no longer (valid|available)/ bad-mailbox
    /mailbox +(is +)?full/ quota-issues
    /banned/ policy-related
</bounce-category-patterns>
```

`<bounce-category-patterns>` uses pattern matching in the same method as
does `<pattern-list>`. describes SMTP pattern lists and
pattern matching in greater detail.

10.7 Combining domains with the same MX record(s) into one queue

10.7.1 Overview

PowerMTA has the ability to group together email domains if they share the same MX
records. This can be useful for controlling the message stream across multiple domains.
This includes any <domain> based controls such as max-msg-rate, max-smtp-out, and
max-connect-rate.

The directive to use is "queue-to", which is defined in the <domain> definitions that you want to group together. The parameter defined within the queue-to directive is the name of the "domain" that the other domains will be rolled up into. Use of the route directive is required. The route directive should point to the root domain to which mail should be delivered.

For example:

```
#
<virtual-mta hotmail-queue>
  <domain hotmail.queue>
    max-smtp-out 10
    route hotmail.com
  </domain>
  <domain hotmail.com>
    queue-to "hotmail.queue"
  </domain>
  <domain hotmail.co.uk>
    queue-to "hotmail.queue"
  </domain>
  <domain live.com>
    queue-to "hotmail.queue"
  </domain>
</virtual-mta>
#
```

With this, all messages queued for this VirtualMTA for any hotmail.com, hotmail.co.uk, or live.com domain will be put in one queue named "hotmail.queue", and treated as such.

Or:

```
#
<virtual-mta client-mta>
    smtp-source-host 1.2.3.4 mail1.yourdomain.com
    <domain [*.]psmtp.com>
        queue-to "psmtp.com"
        max-smtp-out 20
        bounce-after 2d
        max-msg-per-connection 10
    </domain>
</virtual-mta>
```

With this, all messages queued for this VirtualMTA for any psmtp.com domain (e.g. mail9.psmtp.com, someone.psmtp.com, etc.) will be put in the one queue for psmtp.com, and treated as such. This also offers the flexibility to pick and choose which domains/subdomains to roll up into the one domain, or to roll up all domains, by having the queue-to defined within <domain *> in the VirtualMTA. As a note, if you are using

261

a cold VirtualMTA, the directive max-cold-virtual-mta is applied AFTER queue-to is applied.

10.8 Setting recipient priority

10.8.1 Overview

It may be necessary to set a given recipient of set of recipients with a higher or lower priority (e.g. password reset vs. weekly newsletter).   This can be facilitated in PowerMTA by a couple different means. The first is via the <domain> directive "queue-priority". The directive works in the following manner:

```
<domain someplace.com>
  queue-priority 75
</domain>

<virtual-mta newsletter>
  <domain someplaceelse.com>
    queue-priority 30
  </domain>
</virtual-mta>
```

With the above set, all messages going to someplace.com will get a queue-priority of 75 (the default is 50 with a range of 1-100, higher amounts being given higher priority). This will give this set of messages first priority in PowerMTA for connections and delivery. The messages for someplaceelse.com/newsletter will get a lower priority than the average mail, and will go out last.

Another method of setting the message priority is by using a <pattern-list>. The directive works in the following manner:

```
<pattern-list myList>
  rcpt-to /password-reset@customer.com/ recipient-priority=100
  rcpt-to /special@customer.com/ virtual-mta=high, recipient-priority=60
  mail-from /newsletter/ recipient-priority=25
</pattern-list>
```

In the above example, individual emails to password-reset@customer.com will get the highest priority, generic emails to special@customer.com will get a slightly higher than normal priority, and any emails from anything containing newsletter will get a low priority.

If both queue-priority and recipient-priority are used, queue-priority takes precedence. recipient-priority is only used for determining which recipients in a queue a retried first.

262

10.9 SNMP Support

10.9.1 Overview

PowerMTA now supports SNMP for monitoring basic status information.
Currently all the same information from the pmta show status command is what is vailable for monitoring via SNMP. This may be expanded over time to include additional information.

PowerMTA SNMP support is included with the Windows installer. There is a separate package available for Linux from the PowerMTA download page. PowerMTA's SNMP daemon can be used in master mode or as an AgentX subagent (not available on Windows). It automatically detects whether there is a AgentX master running (e.g. Net-SNMP's snmpd). If so it registers itself as a subagent, otherwise it runs in master mode.

As a note, to allow Net-SNMP's snmpd to act as a AgentX master it is required to activate the AgentX support in snmpd (by having a line "master agentx" in snmpd.conf).

The default configuration for PowerMTA's SNMP daemon is located in:

Linux:
    /etc/pmta/pmtasnmpd.conf
Windows:
    <PowerMTA directory>\pmtasnmp.conf

This will allow anybody to read data from it with the "public" community string, which may or may not be appropriate. Since the SNMP daemon is based on Net-SNMP the configuration file is a standard Net-SNMP configuration file and documentation is available from:
http://www.net-snmp.org/docs

The data available through SNMP is described by the MIB which can be found at the following location:

Linux:
    /usr/share/snmp/mibs/PowerMTA-MIB.txt
Windows:
    <PowerMTA directory>\mibs\PowerMTA-MIB.txt

To access the data via SNMP a third party solution is required.
On Linux you have the Net-SNMP utils installed you can test that it works (and see the data available) with the following command:

snmpwalk -v2c -c public localhost PowerMTA-MIB::mta

10.10 High Availability Configuration

Most customers go with a 2 server setup. Typically both servers are running, and if one fails, new mail gets sent to the backup server by the feeding application. If the spool directory is recoverable, it is added as a second spool to the backup, and the service is restarted to pickup this spool, sending all new and old mail.

When the production server is backup and online, new mail is redirected back to the production server, and the hot backup is left running until all mail has been delivered or bounced. After the mail is gone from the hot backup, it is put back into its original single spool state, and is kept idle until the next time it is needed.

10.11 Precached domains support

For senders that have DNS server issues when loading large amounts of domains into PowerMTA in a short period of time, or when maintaining large amounts of domains with short a short DNS TTL, PowerMTA now offers the ability to precache a given subset of the DNS names to ensure the DNS name is always available. This feature is enabled by use of the new, global <dns> tag.

```
<dns>
  precached-domains-file C:\pmta\precached-domains.txt
  precached-max-domains 500
  precached-refresh-interval 5m
</dns>
```

In

The above defined file would look something like the following:

```
hotmail.com
yahoo.com
aol.com
gmail.com
```

PowerMTA would only precache the first 500 domains in the file, refreshing the DNS for these domains every 5 minutes. Use of this directive may overtax a DNS, and may result in being banned from using a public DNS server like Google or OpenDNS. It is highly recommended to use a dedicated, in-house DNS server when using this feature.

10.12 Scheduled Delivery Control

PowerMTA now supports the ability to schedule deliveries via a header. This may be very useful for instances when it takes a long time to build a campaign or if there is a need to have a campaign go out very quickly (e.g. flash sales).

The format is:

```
x-schedule: <start time - 1>/<end time - 1>, <start time - 2>/<end time -2>
```

Here is an example:

```
x-schedule: 2015-05-29 17:01:11 / 2015-05-29 17:30:11, 2015-05-30 17:01:11 / 2015-05-30 17:30:11
```

As many times as you want are allowed, as long as you fold the header so each line is not longer than 1000 characters.

The bounce-after does not apply to such recipients. The schedule overrides the bounce-after and the message is bounce when there are no more schedules to try. In addition, if the queue is in retry mode when the messages are injected, the message delivery will not start at the window start time, but rather message delivery attempts will start when the queue come out of retry mode.

For strict adherence to the schedule start time for large campaigns like flash sales it is required to set the same jobID for all recipients of the given campaign. For best results, customers should ideally use the same schedule for all the recipients in a job, and not mix scheduled and unscheduled recipients in a job. Messages with the combination of Scheduled Delivery Control and a jobID will use the start time defined by the first message for the job injected into the queue.

For example, if the first recipient in the queue for jobID 123 has a start time of 12pm, and the second recipient injected into the queue for the same jobID has a start time of 11am, the 12pm start time will be used for both recipients. Likewise, if the first recipient in the queue for jobID 123 has a start time of 11am, and the second recipient injected into the queue for the same jobID has a start time of 12pm, the 11am start time will be used for both recipients.

10.13 Custom retry intervals

retry-after and backoff-retry-after now take a time interval in addition to the static, defined time. This allows for adding up to 30 different retry periods. This may be useful for domains in which the likelihood of messages being delivered decreases over time. The last defined interval is used until the message bounces. The interval resets on a restart of the PowerMTA server. An example usage would be:

```
<domain example.com>
  retry-after 10m,10m,10m,10m,10m,10m,30m,30m,30m,30m,1h,1h,2h,2h,4h,4h,10h
</domain>
```

In the above example the domain would be tried every 10 minutes for the 6 tries, every 30m for the next 4 tries, every hour for the 2 tries, every 4 hours for the next two tries, and finally, every 10 hours until the message is bounced.

10.14 Address Suppression Lists

Lists can be defined with the new global "<address-list listName>" tag and one or more can be referenced within a <source> with the new "suppression-lists" directive. Addresses in the suppression list are rejected (or turned into bounces, depending on options) during submission. Multiple suppression lists may be used. :

```
<address-list suppression1>
  address foo                           # local part
  address foo@bar.com                   # email address
  address /foo/                         # pattern
</address-list>

<address-list suppression2>
  domain foo.com                        # domain
  address-file /etc/pmta/addresses     # email addresses or
local parts, 1/line
</address-list>

<source 0/0>
  suppression-lists suppression1,suppression2
</source>
```

NOTE: Defining more than 500 suppression list patterns is not recommended at this time.

If the <source> directive accept-invalid-recipients is set to "no", then the message is rejected at the time of submission (i.e., while processing the RCPT command).

If the <source> directive accept-invalid-recipients is set to "yes", then the message is queued and later bounced with generation of a bounce record with details such as:

```
dsnStatus: 5.7.1 (delivery not authorized)
dsnDiag: smtp;550 5.7.1 recipient is in suppression list "a1": <foo@bar.com>
```

For BSMTP pickup files, If XACK is set to "on", then the message is rejected while the file is picked up (i.e., while processing the RCPT command). This is logged in the PowerMTA log file just like any other pickup submission error. If XACK is set to "off",

then the file is picked up and the message is later bounced with a bounce record that looks similar to SMTP submission case above.

For non-BSMTP pickup files, the message is rejected while the file is picked up (i.e., when processing the x-receiver header). This is logged in the PowerMTA log file just like any other pickup submission error. There is no mechanism currently to submit the message successfully and/or to get a bounce record in this case.

10.15 MX Rollup List

Allows one to define "rollup" queues based on the MX records of the recipient domain, in order to consolidate separate but related recipient domains into one queue. Messages moved into the roll up queue are handled as if they were one recipient domain with regards to rate limiting, connection caps, etc....   For example, since msn.com, live.com and hotmail.com are all handled by the same exact MXs/gateway, it makes sense to roll these up into one queue (hotmail.rollup for example) vs. being handled separately by PowerMTA. This feature works very well for the large mailbox providers that provide corporate hosting, for filtering/antispam cloud providers, or for large hosting providers in general that manage corporate email for tens or hundreds of thousands of various domains.

To configure this feature, you need to define the MX record that maps to each rollup queue name along with the name of the rollup queue, which must end in ".rollup". More than one MX can be mapped to the same rollup queue, for example:

```
<mx-rollup-list>
  mx mx1.hotmail.com hotmail.rollup
  mx mx2.hotmail.com hotmail.rollup
  mx mx3.hotmail.com hotmail.rollup
  mx mx4.hotmail.com hotmail.rollup
  mx smtp.secureserver.net secureserver.net.rollup
</mx-rollup-list>
```

NOTE that only the lowest preference (highest priority) MX records should be defined with this feature, for that is all that we key on for this feature.

Wildcards are also supported for the MX records, for example:

```
<mx-rollup-list>
  mx *.hotmail.com hotmail.rollup
</mx-rollup-list>
```

which would then automatically include all matching lowest preference MX records. Any recipient domain whose lowest preference MXs match the wildcard would be automatically placed in the hotmail.rollup queue, and which all messages would be treated as if to a single recipient domain.

No match occurs when a rollup queue defined in the <mx-rollup-list> has more lowest preference MXs configured to it than what is returned by DNS.

For rollup queues in which the MX records are explicitly defined (no wildcards) in the rollup list, PowerMTA will automatically use/look up the specific defined MXs in DNS for routing information for the domains in the rollup queue.

For rollup queues in which a wildcard is used, one **must** explicitly define the smtp-hosts (or route) directive within the domain definition for the rollup queue along with the appropriate MX or A records, since it is impossible for PowerMTA to infer the proper MXs to use for all domains in the rollup queue (in contrast to when they are explicitly defined).

Port25 recommends always explicitly defining the rollup queue within its own domain definition in order to explicitly set the delivery controls you want to achieve for the rollup queue.

For clarification, the table below provides various scenarios and end results based on the following MX rollup list definitions and defined DNS results.

Example configuration settings for MX Rollup list to be used for chart below:

```
<mx-rollup-list>
  mx mx1.domain.com domain.rollup
  mx mx2.domain.com domain.rollup
  mx mx1.domain2.com domain2.rollup
  mx *.mail.domain3.com domain3.rollup
</mx-rollup-list>
```

The below table lists various examples of how this would work. For all examples the recipient domain would be domain.com.

MX Records in DNS and Preference	Rollup Queue used	Equal MXs used by PowerMTA for Queue	Explanation
mx1.domain.com 10 mx2.domain.com 10	domain.rollup	mx1.domain.com mx2.domain.com	MXs defined for rollup queue are present in domain's DNS results.

MX records	Rollup queue	Hosts used	Notes
mx1.domain.com 10 mx2.domain.com 20	N/A – rollup would not take place	**N/A**	Rollup queue has more lowest-preference MXs defined than what DNS returns.
mx1.domain.com 10 mx2.domain.com 10 mx3.domain.com 10	domain.rollup	mx1.domain.com mx2.domain.com	MXs defined for rollup queue are present in domain's DNS results, mx3.domain.com is not used by PowerMTA for delivery.
mx1.domain.com 10 mx2.domain.com 10 mx3.domain.com 10	domain.rollup	mx1.domain.com mx2.domain.com	MXs defined for rollup queue are present in domain's DNS results, mx3.domain.com is not used by PowerMTA for delivery.
mx1.domain.com 10	N/A – rollup would not take place	**N/A**	Rollup queue has more lowest preference MXs defined than what DNS returns.
mx1.domain2.com 10 mx2.domain2.com 10	domain2.rollup	mx1.domain2.com	MXs defined for rollup queue are present in domain's DNS results, mx2.domain2.com is not used by PowerMTA for delivery.
mx1.domain2.com 10 mx1.domain2.com 20	domain2.rollup	mx1.domain2.com	MXs defined for rollup queue are present in domain's DNS results.
mx1.mail.domain.com 10	N/A – rollup would not take place	**N/A**	DNS results do not match any defined rollup queue.
mx1.mail.domain3.com 10 mx2.mail.domain3.com 10 mx3.mail.domain3.com 20 mx4.mail.domain3.com 20	domain3.rollup	Hosts explicitly defined in smtp-hosts directive for the rollup queue.	MXs cannot be auto inferred when using wildcards, as such, one must explicitly define the hosts that PowerMTA must use when using wildcards.

11. The Accounting and Statistics

11.1 Introduction

The accounting file can be written in a CSV format or streamed to pipe. Some of the combinations (but not all) include:

- CSV format only
- Streamed to pipe only
- Both CSV and pipe format

11.2 The Accounting File

For efficiency and ease of use, PowerMTA writes accounting information to a separate file rather than to its logging file. Accounting is enabled in the initial configuration, so unless you disabled it you should find the accounting file(s) in the same directory as the logging file(s).

Since PowerMTA will write detailed information to the accounting file for each message and report delivered, it will also increase overhead and such could impact throughput performance with larger workloads. If the accounting file is of no interest, you can disable it in the configuration file, however please note that the information included in the accounting file for each message is not readily available anywhere else in the solution.

For ease of use, the accounting file is a CSV (Comma Separated Value) file.

There are 8 types of records available:

- *Successful delivered* records which provide the data for all messages successfully transferred to the destination domain mail servers.
- *Bounced* records which provide the data for all reports PowerMTA generated and delivered, including bounces for messages that PowerMTA could not deliver.
- *Transient error & transient queue error* records which show each unsuccessful delivery attempt of an individual message. These errors can be both for individual messages, or queue wide.
- *Inbound recipients* records which show each successful submission of an individual message to PowerMTA.
- *Feedback loop* records can be logged if PowerMTA has been configured with a <feedback-loop-processor>
- *Remote bounce* records can be logged if PowerMTA has been configured with a <bounce-processor>
- *Remote Status* records can be logged if PowerMTA has been configured with a <bounce-processor>

Note that the "bounced" records may not include records for *all* bounces because messages routed through intermediary, relaying mailers may still bounce on their way to the final destination. These remote bounces (i.e. asynchronous bounces sent back by other mail servers) are recorded as successful transfers since they were successfully transferred initially, and at that point are completely out of the control of PowerMTA. All records are written to the accounting file at the time the message or report is delivered, so they will be ordered by delivery time.

11.2.1 Accounting records

PowerMTA writes an accounting record per recipient delivered. These various records can be stored in all in one file, or in separate files by record type. The CSV field `type` is used to determine one of four possible values for the file. The various types are "d" for delivered, "b" for bounced, or "t" for transient, "tq" for transient-queue, and "r" for received. By default, only "`d`" and "`b`" records are recorded.

11.2.2 "Successful delivered" Records

The following are all the default fields for the CSV file when using "d" or "delivery" as the `record` type.

CSV Accounting File (record-fields)	Description
type	Type of record ("d" or "delivery")
timeLogged	Time the record was delivered and logged to the accounting file
timeQueued	Time message was queued to disk
orig	originator (from MAIL FROM:<x>)
rcpt	recipient (RCPT TO:<x>) being reported
orcpt	original recipient (from RCPT TO:... ORCPT=x), immediately following the recipient to which it refers
dsnAction	DSN action for the recipient to which it refers (Relayed means the message was passed on to another MTA, one that doesn't support the DSN extensions. Delivered means the MTA that issued the report actually did the final delivery.)
dsnStatus	DSN status for the recipient to which it refers
dsnDiag	DSN diagnostic string for the recipient to which it refers
dsnMta	DSN remote MTA for the recipient to which it refers
srcType	source type from which the message was received, either 'api' or 'smtp'
srcMta	source from which the message was received. the MTA name (from the HELO/EHLO command) for messages received through SMTP

`dlvType`	delivery method, one of "`smtp`", "`pipe`", "`discard`", or "`file`"
`dlvSourceIp`	local IP address PowerMTA used for delivery
`dlvDestinationIp`	IP address of the mailer to which the report/message was delivered
`dlvEsmtpAvailable`	SMTP extensions supported by receiving mailer
`dlvSize`	report/message size in bytes
`vmta`	VirtualMTA selected for this message, if any
`jobId`	job ID for the message, if any
`envId`	envelope Id, from `MAIL FROM:... ENVID=`x (present only if specified)
`queue`	domain/vmta used for delivery (remote status only)
`vmtaPool`	VirtualMTA pool selected for this message, if any (remote status only)
`header_XXX`	for each custom header included where XXX is the name of the header, if any
`userString`	String added to accounting file as defined by the <acct-file> directive user-string.

Sample accounting entry for a delivery record:

```
d,1191435989,1191435961,,,testfrom@port25.com,testto@port25.com,,relayed,2.0.0
(success),, [192.168.0.10]
(192.168.0.10),success,api,,smtp,10.25.25.211,10.25.25.20,,316,vmta0,0,0
```

11.2.3 "Bounced" Records

The following are all the default fields for the CSV file when using "b" or "bounced" as the `record` type.

CSV Accounting File (`record-fields`)	Description
`type`	Type of record ("b" or "bounce")
`timeLogged`	Time the record was delivered and logged to the accounting file

272

`timeQueued`	Time message was queued to disk
`orig`	originator (from MAIL FROM:<*x*>)
`rcpt`	recipient (RCPT TO:<*x*>) being reported
`orcpt`	original recipient (from RCPT TO:... ORCPT=*x*), immediately following the recipient to which it refers
`dsnAction`	DSN action for the recipient to which it refers
`dsnStatus`	DSN status for the recipient to which it refers
`dsnDiag`	DSN diagnostic string for the recipient to which it refers
`dsnMta`	DSN remote MTA for the recipient to which it refers
`bounceCat`	likely category of the bounce (see Section 1.5), following the recipient which it refers
`srcType`	source type from which the message was received, either 'api' or 'smtp'
`srcMta`	No data present in Bounce Record
`dlvType`	No data present in Bounce Record
`dlvSourceIp`	No data present in Bounce Record
`dlvDestinationIp`	No data present in Bounce Record
`dlvEsmtpAvailable`	No data present in Bounce Record
`dlvSize`	Deprecated. No data present in Bounce Record
`vmta`	VirtualMTA selected for this message, if any
`jobId`	job ID for the message, if any
`envId`	envelope Id, from `MAIL FROM:... ENVID=`*x* (present only if specified)
`queue`	domain/vmta used for delivery (remote status only)
`vmtaPool`	VirtualMTA pool selected for this message, if any (remote status only)
`header_XXX`	for each custom header included where XXX is the name of the header, if any
`userString`	String added to accounting file as defined by the <acct-file> directive user-string.

Sample accounting entry for a bounce record:

```
b,1191424958,1191424481,,,testfrom@port25.com,testto@port25.com,,failed,4.4.1 (no answer
from host),,[192.168.0.10] (192.168.0.10),no-answer-from-host,api,,,,,,,vmta0,0,0
```

11.2.4 "Transient error" Records

There are times when sending emails may fail. To help identify these issues, PowerMTA has the ability to show transient errors per message. This logging is handled with the record type "t". The fields available are as follows:

CSV Accounting File (record-fields)	Description
type	Type of record ("t" or "transient")
timeLogged	Time the record was delivered and logged to the accounting file
timeQueued	Time message was queued to disk
orig	originator (from MAIL FROM:<x>)
rcpt	recipient (RCPT TO:<x>) being reported
orcpt	original recipient (from RCPT TO:... ORCPT=x), immediately following the recipient to which it refers
dsnAction	DSN action for the recipient to which it refers
dsnStatus	DSN status for the recipient to which it refers
dsnDiag	DSN diagnostic string for the recipient to which it refers
dsnMta	DSN remote MTA for the recipient to which it refers
bounceCat	likely category of the bounce (see Section 1.5), following the recipient which it refers
srcType	source type from which the message was received, either 'api' or 'smtp'
srcMta	source from which the message was received. the MTA name (from the HELO/EHLO command) for messages received through SMTP
dlvType	delivery method, one of "smtp", "pipe", "discard", or "file"
dlvSourceIp	local IP address PowerMTA used for delivery

274

`dlvDestinationIp`	IP address of the mailer to which the report/message was delivered
`dlvEsmtpAvailable`	SMTP extensions supported by receiving mailer
`dlvSize`	Deprecated. report/message size in bytes
`vmta`	VirtualMTA selected for this message, if any
`jobId`	job ID for the message, if any
`envId`	envelope Id, from `MAIL FROM:... ENVID=`*x* (present only if specified)
`queue`	domain/vmta used for delivery (remote status only)
`vmtaPool`	VirtualMTA pool selected for this message, if any (remote status only)
`header_XXX`	for each custom header included where XXX is the name of the header, if any
`userString`	String added to accounting file as defined by the <acct-file> directive user-string.

Sample accounting entry for a transient record:

```
t,1188490693,1188481072,,,support@port25.com,test@port25.com,,failed,
4.0.0 (undefined status),mailbox unavailable,,
other,api,,,,,,,vmta0,6,0
```

11.2.5 "Transient Queue" Records

There are times when sending emails that an entire given queue may fail. It may be problems with a particular outbound IP address, or maybe one domain is currently offline. To help identify these issues, PowerMTA has not only the ability to show transient recipient errors per message, but also the same information on a per queue basis. This logging is handled with the record type "tq". The records when using "tq" are slightly different than those mentioned above. The fields available are as follows:

CSV Accounting File (`record-fields`)	Description

`type`	Type of record ("tq" or "transient queue")
`timeLogged`	current time
`queue`	queue name
`vmta`	VirtualMTA name
`dsnStatus`	DSN status code for failure
`dsnDiag`	DSN diagnostic string for failure
`dsnMta`	remote MTA, if applicable
`dlvType`	delivery type
`dlvSourceIp`	IP address from which the connection was made/attempted
`dlvDestinationIp`	IP address to which the connection as made/attempted
`userString`	String added to accounting file as defined by the <acct-file> directive user-string.

Sample accounting entry for a transient queue record:

```
tq,1191436554,port25.com/vmta0,,vmta0,4.4.1 (no answer from
host),port0.com,[192.168.0.10] (192.168.0.10),smtp,192.168.0.11,192.168.0.10
```

If the record type "tq" is used for the same CSV file as "d", "b", or "t" all fields from both tables will be included, possibly resulting in many empty data fields.

11.2.6 "Received" Records

PowerMTA can log messages in the accounting file during the injection process. This logging is handled with the record type "r". The records when using "r" are slightly different than those mentioned above. The fields available are as follows:

CSV Accounting File (record-fields)	Description
type	Type of record ("r" or "received")
timeLogged	current time
orig	originator (from MAIL FROM:<x>)
rcpt	recipient (RCPT TO:<x>) being reported
orcpt	original recipient (from RCPT TO:... ORCPT=x), immediately following the recipient to which it refers
srcType	source type from which the message was received, either 'api' or 'smtp'
srcMta	source from which the message was received. the MTA name (from the HELO/EHLO command) for messages received through SMTP
rcvSourceIp	IP address of the server connecting to PowerMTA
rcvDestinationIp	IP address PowerMTA used to handle the connection
vmta	VirtualMTA selected for this message, if any
jobid	job ID for the message, if any
envid	envelope Id, from MAIL FROM:... ENVID=x (present only if specified)
header_XXX	for each custom header included where XXX is the name of the header, if any
userString	String added to accounting file as defined by the <acct-file> directive user-string.

Sample accounting entry for a received record:

```
r,2010-04-17 16:19:57-0400,test@port25.com,test@port25.com,,smtp,tesFeeder
(192.168.25.56),192.168.25.56,192.168.25.11,vmta1,job2,envid1234
```

If the record type "r" is used for the same CSV file as "d", "b", or "t" all fields from both tables will be included, possibly resulting in many empty data fields.

11.2.7 "Remote Bounce" & "Remote Status" Records

PowerMTA can log remote (asynchrouns) bounces in the accounting file and remote status reports. This logging is handled with the record type "rb" for remote bounce and "rs" for reporting other kinds of remote status (used for DSN positive and "delayed" status notifications). The records when using "rb" or "rs" are slightly different than those mentioned above. The fields available are as follows:

CSV Accounting File (record-fields)	Description
type	Type of record "rb" or "rs"
timeLogged	current time
timeQueued	time message was queued to disk (remote status only)
orig	originator address (of original email)
rcpt	recipient address (of original email)
orcpt	original recipient (from RCPT TO:... ORCPT=*x*), immediately following the recipient to which it refers
dsnAction	DSN action (remote status only)
dsnStatus	DSN status
dsnDiag	DSN diagnostic
dsnMta	DSN remote MTA
dsnReportingMta	DSN reporting MTA (remote bounce only)
bounceCat	bounce category
srcType	source type from which the message was received, either 'api' or 'smtp'
srcMta	source from which the message was received. the MTA name (from the HELO/EHLO command) for messages received through SMTP
srcMta	source from which the message was received. the MTA name (from the HELO/EHLO command) for messages received through SMTP (remote status only)
dlvType	delivery type (remote status only)
dlvSourceIp	IP address from which the connection was made/attempted (remote status only)
dlvDestinationIp	IP address to which the connection as made/attempted (remote status only)
dlvEsmtpAvailable	SMTP extensions supported by receiving mailer (remote status only)
dlvSize	report/message size in bytes (remote status only)

278

vmta	VirtualMTA selected for this message, if any (remote status only)
jobId	job ID for the message, if any (remote status only)
envid	envelope Id, from MAIL FROM:... ENVID=*x* (present only if specified)
queue	domain/vmta used for delivery (remote status only)
vmtaPool	VirtualMTA pool selected for this message, if any (remote status only)
header_<name>	header extracted from original email

Sample accounting entry for a received record:

```
rb,2010-12-01 14:24:44-0500,bounce@bounce.com,lksjflkj@somedomain.com,,5.1.1 (bad destination
mailbox address),"smtp;550 5.1.1 no such local recipient: <lksjflkj@somedomain.com> in ""RCPT
TO:<lksjflkj@somedomain.com>""",mail.remotedomain.com (169.3.169.30),my.local.server.com,bad-
mailbox,smtp,my.local.server.com (192.168.0.150),
```

If the record type "rb" or "rs" is used for the same CSV file as "d", "b", or "t" all fields from both tables will be included, possibly resulting in many empty data fields.

11.2.8 "Feedback Loop" Records

PowerMTA can log feedback loop reports in the accounting file. This logging is handled with the record type "f". The records when using "f" are slightly different than those mentioned above. The fields available are as follows:

CSV Accounting File (record-fields)	Description

type	Type of record "f"
timeLogged	Current time
repSourceIp	the IP of PowerMTA that received the message
format	"arf" or "jmrp"
userAgent	name & version of the program that generated the report
envid	envelope ID (of original email)
orig	originator address (of original email)
rcpt	recipient address (of original email)
reportingMTA	the name of the MTA generating this feedback report
dlvSourceIp	address of MTA from which the message was received
reportedDomain	one or more domain names reported relevant
header_From	from header (of original email)
header_Return-Path	return-path header (of original email)
header_X-job	x-job header (of original email)
header_Subject	Subject header (of original email)
header_<name>	header extracted from original email
arf_<name>	field from ARF feedback-report
reportedDomain	the domain about which the feedback report was generated
feedbackType	mapped to ARF's Feedback-Type field

Sample accounting entry for a FBL record:

```
f,2010-04-12 11:40:37-0400,arf,AOL SComp,,,,mail1.domain.com,169.63.151.30
```

If the record type "f" is used for the same CSV file as "d", "b", or "t" all fields from both tables will be included, possibly resulting in many empty data fields.

11.2.9 Encoding

As much as possible, standard CSV encodings are used for all fields, with most characters being as-is in the output. Time stamps are normally formatted as "MM/DD/YYYY HH:MM:SS+UTCoffset" but can be changed to UNIX time_t, i.e., the number of seconds since midnight on January 1, 1970.

The format of the file is CSV (comma-separated values), UTF-8 encoded, as described in RFC 4180, with a header line at the beginning of the file describing what field is in which column. Here is the format from RFC 4180 in short: columns are separated by commas and records terminated with CRLF. Values may be quoted with double quotes (""), and must be quoted if its value contains a comma, a double quote or CRLF. A double quote occurring within double quotes is repeated, like in:

```
two inches,"2"""
```

To accommodate writing various record types in the same file, the first column is always the record type. If an accounting file is configured (or defaulted) to have multiple record types, all the possible columns from the selected records are included. For example, if a file was configured as:

```
records d, b
record-fields delivery timeLogged,orig,rcpt,dlvSourceIp
record-fields bounce timeLogged,orig,rcpt,bounceCat
```

the resulting file (including a few record of each type) might look like:

```
type,timeLogged,orig,rcpt,dlvSourceIp,bounceCat
d,1178807903,orig@port25.com,user1@domain.com,10.0.0.1,
d,1178807905,orig@port25.com,user2@domain.com,10.0.0.1,
b,1178807915,orig@port25.com,user3@domain.com,,policy-related
b,1178807915,orig@port25.com,user4@domain.com,,policy-related
```

Note that the dlvSourceIp field was omitted (empty field at the end) for the 'b' record, and that the bounceCat was omitted in the 'd' record, because they were not configured.

11.3 Configuring the accounting file

All new installs in which a previous version of PowerMTA is not present will use the CSV format.

Use of the new accounting file format requires the `<acct-file>` tag. The tag takes one or more parameters. If passing one parameter, it can specify the location of the CSV file. If passing more than one, it can specify the location of the executable program to which PowerMTA should pipe the accounting records. If using pipe, the first parameter will need to be preceded by a "|". It may be required to quote the acct file path if it contains spaces (like <acct-file "c:\my stuff\acctproc.exe">) or if arguments are passed (like <acct-file "/bin/foo.pl -x -a -z">) Examples are listed in the <u>section 1.3.2</u>.

In the current pipe implementation, if PowerMTA runs into a problem writing to the pipe, it gives up on it and won't retry either until the next "pmta reload" command is run, or after a certain predefined time limit. With this, it will be necessary for customers to configure PowerMTA to create a corresponding accounting data file as well, in order to have the data to recover, if there are problems with the customer's pipe application.

In the event of a broken pipe, PowerMTA writes:
   Error TYPE "CMD": DETAILS
to the logging file, where TYPE is one of "starting", "writing to" or "executing", CMD is the configured command, and DETAILS is a string with details on what went wrong. Each time a pipe is started, the first line written to it is a headers line with the various fields (like in the CSV files). It is only re-started upon a "pmta reload".

11.3.1 Directives

The User's Guide covers the various directives that can be used with `<acct-file>`, but they are included here for your convenience.

delete-after

> **Scope:** acct-file
>
> **Type:** {#d|never}
>
> **Attributes:** optional
>
> **Default:** 8d (8 days)

> This directive tells PowerMTA how long to keep the accounting file before deleting it from the server. "delete-after" only applies if "move-to" isn't active" (and move-to must point to a different location than the directory where the file was originally created).

move-to

> **Scope:** acct-file
>
> **Type:** directory
>
> **Attributes:** optional
>
> **Default:** files not moved

> This directive tells PowerMTA where to move the accounting file when using `move-interval` or `max-size`. Directory must be on the same volume as the accounting files. Moving across a network is not supported.

move-interval

> **Scope:** acct-file
>
> **Type:** time interval
>
> **Attributes:** optional
>
> **Default:** 1d

> This directive tells PowerMTA at what time interval to move the accounting file and start a new one.

`max-size`

> **Scope:** acct-file
> **Type:** {n{B|K|M|G|T}
> **Attributes:** optional
> **Default:** 250M

> The max size of the accounting file in Bytes, Kilobytes, Megabytes, Gigabytes, or Terabytes before PowerMTA moves the accounting file and starts a new one. Minimum size is 1M (1000000B & 1000K) & Maximum size is 1T. This directive applies both with and without "`move-to`". If "`move-to`" is not specified, a new file is started in place (with a higher –NNNN in the file name). The max value is 8388607T.

`retry-interval`

> **Scope:** acct-file
> **Type:** time interval
> **Attributes:** optional
> **Default:** 5m

> Retries the pipe program in a regular interval in case of a failure. This should only be used when using pipe to write an accounting file.

`user-string`

> **Scope:** acct-file
> **Type:** string
> **Attributes:** optional
> **Default:**

> Specifies a custom string to be stored along with other MTA information in theaccounting file.

`write-timeout`

      Scope: acct-file

      Type: time interval

      Attributes: optional

      Default: 1m

Allows for setting write timeouts in accounting pipes, so that an unresponsive accounting application doesn't hold up PowerMTA indefinitely. Only used when pipeing accounting data to another file.

`iso-times`

      Scope: acct-file

      Type: boolean

      Attributes: optional

      Default: yes

Allows requesting that PowerMTA formats time stamps in the accounting file (or pipe) in ISO format.

```
records
```
      Scope: acct-file

      Type: one or more comma-separated record types

      Attributes: optional

      Default: d,b

This directive tells PowerMTA which record types to include in the accounting file. There are 8 record types: d, b, t, tq, r, f, rb, & rs. It may only be defined once per <acct-file> with multiple types separated by commas.

- "d" is for delivered records and will include all messages that were successfully delivered. "delivery" can be used in place of "d".

- "b" is for bounced records and will include all locally generated bounces (synchronous bounces). Note that bounces generated by remote mail servers (asynchronous bounces) will not be included. "bounce" can be used in place of "b".

- "t" is for transient recipient level errors only (such as mailbox full). "transient" can be used in place of "t".

- "tq" is for queue wide transient errors and will include information about delivery attempts for a given domain/vmta. "transient-queue" can be used in place of "t".

- "r" is for inbound records and will include information about messages submitted to PowerMTA for delivery. "receipt" can be used in place of "r".

- "f" is for feedback loop emails if PowerMTA is configured to look for feedback loop emails with the <feedback-loop-processor>. "feedback-loop" may be used in place of "f". See Section 12.7 for more information.

- "rb" is for remote bounce emails if PowerMTA is configured to look for remote bounce emails with the <bounce-processor>. "remote-bounce" may be used in place of "rb". See Section 12.6 for more information.

- "rs" is for remote status emails if PowerMTA is configured to look for remote status emails with the <bounce-processor>. "remote-status" may be used in place of "rs". See Section 12.6 for more information.

Due to the fact that there can be many delivery attempts for one message, it is suggested that type "t" and/or "tq" should only be used when needed (debugging, troubleshooting, etc.) as it may cause large file sizes. It is also recommended when using this setting to have the "t" or "tq" type records written to their own file, as so not to cause unwanted data in the primary accounting file.

```
record-fields
```
Scope: acct-file

Type: record type followed by a comma-separated list of field names

Attributes: optional

Default: *

This directive specifies the fields to be included in the accounting file. By default all are included. To include only a subset of all the fields, simply add the desired fields in a comma separated list. To include a custom header, add the record field header_XXX, where XXX is the name of the custom field you wish to log. For example, to log all fields and a custom header, the directive would look like this:

```
record-fields delivery *,header_Message-Id
```

To include only a subset of the fields, list the required fields similar to the following:

```
record-fields bounce timeLogged,orig,rcpt,dlvSourceIp,vmta
```

To exclude fields from the default, you can use an "!" to exclude the unwanted fields. For example, to include all fields except orcpt you would use:

```
record-fields d *,!orcpt
record-fields b *,!orcpt
```

With the mailing industry moving more to a secure and encrypted sending model, it is necessary to track if a given message was delivered over a secure socket, and if so, what protocol and cipher were used. To this end, users can now choose to have PowerMTA add this information into the CSV accounting file by defining the fields dlvTlsProtocol & dlvTlsCipher in the record-fields directive. For example:

```
<acct-file log\acct.csv>
    records d, b
    record-fields d *, dlvTlsProtocol, dlvTlsCipher
    record-fields b *, dlvTlsProtocol, dlvTlsCipher
</acct-file>
```

In the event that the authenticated user who submitted a message needs to be tracked, this can be done with the rcvSmtpUser field.

```
<acct-file log\acct.csv>
    records d, b
    record-fields d *, rcvSmtpUser
    record-fields b *, rcvSmtpUser
</acct-file>
```

286

world-readable

> **Scope:** acct-file
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** no

This directive tells PowerMTA what read permissions to set on the accounting file.

count-moved-records

> **Scope:** acct-file
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** false

This directive tells PowerMTA whether or not to count the records in CSV files moved. The record counts are written to the logging file along with the message that indicates that the file was moved.

sync

> **Scope:** acct-file
>
> **Type:** boolean
>
> **Attributes:** optional
>
> **Default:** no

Determines whether the OS' buffers for the accounting file are flushed regularly. This helps ensure that the accounting data isn't corrupted in case of a system crash.

11.3.2 Examples

The accounting file is configured with the <acct-file> type directive with the date added as part of the file name. For example <acct-file log/acct.csv> yields a file name like /pmta/log/acct-2007-09-05-0000.csv. By default on a new installation, the settings will look similar to the following:

```
<acct-file log\acct.csv>
#   move-to c:\myapp\pmta-acct   # configure as fit for your application
    move-interval 1d
    max-size 250M # MB
</acct-file>
```

287

The data may also be written to pipe. This can be useful if a custom application is created in which the data would be written to a database in real time. PowerMTA instantiates the script only once at startup and then keeps passing data to the pipe as and when available. An example would look similar to the following:

```
<acct-file |C:\pmta\bin\customCode.exe --thisOption --thatOption>
</acct-file>
```

Multiple files can be used if needed. For example, it may be more convenient to have all bounces and deliveries in one file and transient reports in another file. An example of such would be similar to the following:

```
<acct-file log\acct-bd.csv>
   records b,d
   record-fields d *,!orcpt,header_myCustomHeader
   record-fields b *,!orcpt
</acct-file>

<acct-file log\acct-t.csv>
   records t,tq
</acct-file>
```

11.4 The `pmtastats` Accounting Statistics Application

PowerMTA comes with an accounting file parsing application, `pmtastats`, that calculates delivery throughput and general traffic statistics both on a global basis and/or on a per campaign or mailing basis. This application was historically known as "acctstats" but renamed to "pmtastats" starting in v3.5, given the enhanced functionality added to this application. If you want to generate the historical "acctstats" output with "pmtastats", simply use the "--acctstats" flag when running the application. For example:

```
pmtastats --acctstats acct.csv
```

You can use `pmtastats` with either binary or CSV-encoded accounting files: it automatically detects the file format and processes it accordingly.

If you wish to run the parsing application over more than one accounting file (for example, if the accounting file was rotated while sending a campaign), simply specify the various files within the same command. For example:

```
pmtastats [global options] [statistic [options] ..] acct.dat acct-1.dat
```

11.4.1 `pmtastats` Options

`pmtastats` has the following options, as shown by the `--help` output:

288

```
PowerMTA(TM) v3.3 statistics program
Copyright(c) 1999-2007, Port25 Solutions, Inc.  All Rights Reserved.

Usage: pmtastats [global options] [statistic [options] ...] file ...

One or more statistics can be specified, each with its own options.  Use with
one or more files, in either binary, CSV or XML format.  '-' can be used for
reading from standard input.

examples:
     pmtastats vmta-summary vmta-time-breakdown acct*.*
     pmtastats bounce-categories --bounceCat all acct*.*
     pmtastats --vmta vmta1 top-rates delivery-times acct*.*
     pmtastats --output html vmta-time-breakdown acct*.* > report.html

global options:
    --help
        display this help and exit
    --version
        display version information and exit
    --quiet
        do not display progress information
    --output <format>
        selects the output format (text, html; default: text)
    --envId <id>
    --from <originator>
    --toDomain <domain>
    --jobId <id>
    --vmta <vmta>
    --rcvSmtpUser <username>
    --header <name> <content>
        filter input: use only records whose envelope ID, originator,
        recipient domain, job ID, VirtualMTA, or the given header
        starts with the given string, respectively
        (default: no filtering)
    --last DdHhMmSs
        filter input: use only records written in the last D days, H hours,
        M minutes and S seconds.  Each part of the time specification can
        be omitted, such as in "3h15m" for 3 hours, 15 minutes.
        (default: no filtering)
    --byEnvId
    --byJobId
    --byFrom
    --byVmta
    --byHeader <header>
        statistics split by envelope ID, job ID, originator, VirtualMTA, or a
        given header, respectively.
        Only one of these can be used.
        For --byHeader, specify the header.
        (default: statistics not split)
    --splitLimit <num>
       set the maximum number of statistics a statistic can be split into
       (default: 5000)
    --topDomains <num>
        display top <num> domains (passed on to all statistics)

"bounce-categories" statistic
Shows most frequent bounce categories.
Options:
    --topCategories <num>
        display top <num> bounce categories (default: 10)
    --bounceCat <option>
        selects categories for detailed display.  Possible options are:
            <num> - top <num> categories
            <category> - category name
            'all' - all available categories
        (default: spam-related)
```

289

```
    --byDsn <num>
        display bounce category detail for the top <num> DSN codes (default: 10)
    --byDomain <num>
        display bounce category detail for the top <num> domains (default: 10)
    --byDsnAndDomain <num>
        display bounce category detail for the top <num> DSN code / domain
        combinations (default: 10)

"delivery-times" statistic
Shows times and recipients / time from queueing to delivery.
Options:
    (none)

"just-delivery-times" statistic
Time just for delivery.
Options:
    (none)

"message-counts" statistic
Total counts of recipients and volume for messages and bounces.
Options:
    (none)

"top-bounce" statistic
Top bounce domains, aggregated by DSN code (reason).
Options:
    --topDomains <num>
        display bounce statistic for top <num> domains (default: 10)
    --topStatusCodes <num>
        display bounce statistics for top <num> status codes (default: 10)

"top-domains" statistic
Domains with the most recipients, bounces, and delivery times.
Options:
    --topDomains <num>
        display top <num> domains (default: 10)
    --noDeliveryTimes
        omits list of domains with the highest average delivery times
        (default: includes it)

"top-rates" statistic
Data transfer and recipient rates over time.
Options:
    (none)

"vmta-summary" statistic
High-level VirtualMTA traffic summary.
Options:
    --byDate
        split records by date (default: no splitting done)

"vmta-time-breakdown" statistic
Dailiy and hourly recipients and bounces for each VirtualMTA.
Options:
    --noHourly
        do not include hourly breakdown (default: include it)

"vmta-top-bounce" statistic
Bounces for individual VirtualMTAs.
Options:
    --topBounced <num>
        show top <num> domains by bounces (default: 20)
    --byDate
        split records by date (default: no splitting done)

shortcut options:
    --all
```

290

```
        create all available statistics with default options
    --acctstats
        create statistics similar to what acctstats did
```

Note: On Windows, you can also use the "/" character instead of "--" (but "--" works on all platforms).

The following is a list of the various fields available, and which reports return those fields.

	bounce-categories	delivery-times	just-delivery-times	message-counts	top-bounce	top-domains	top-rates	vmta-summary	vmta-time-breakdown	vmta-top-bounce
bounceCat	*									
dlvSourceIp								*	*	*
dsnDiag	*			*	*			*	*	*
dsnStatus	*				*					
envId										
header_<xyz>										
jobId										
nRcpt	*	*			*		*	*	*	*
orig										
rcpt				*	*	*				*
size				*			*			
timeLogged		*				*		*	*	*
timeQueued		*	*			*				
vmta								*	*	*

291

11.4.2 `pmtastats` Report Breakdown

Given the various reports available, different data will or will not be included in each report based on the options used. Here is a sample of the reports available and their output. Multiple reports can be run at the same time.

11.4.2.1 Time Frame

```
    Time Frame:
                                       first               last
                                       ------------------- -------------------
                        received       2007-01-15 14:19:11 2007-01-17 00:42:01
                        delivered/bounced 2007-01-16 03:30:45 2007-01-17 03:15:46
```

This section is always generated for each report type, and shows the first and the last receipt time stamps and the first and the last delivery time stamps of the applicable messages. The scope of the output is based on the type of report specified, i.e., global or by originator or by envelope ID, and such can be used to calculate a campaign duration time. In this example, the mailing duration would be 13 hours 56 minutes and 35 seconds, which is the difference between when the first message was received (2007-01-15 14:19:11) and the last message was delivered (2007-01-17 03:15:46).

11.4.2.2 Bounce Categories

```
Total Bounce Breakdown By Category:

   Bounce reports (across 14 categories):

   total bounced  5.X.X bounces  4.X.X bounces
   ------------   ------------   ------------
         94,457         89,678          4,779

   bounced  %bounced  bounce category      description
   -------  --------  ------------------   -----------
    42,600    45.1%   other                messages rejected due to other reasons, 4.X.X or 5.X.X error
    27,175    28.8%   spam-related         messages refused or blocked due to spam related reasons, 5.X.X error
    13,045    13.8%   bad-mailbox          messages rejected due to bad, invalid, or non-existent recipient addresses, 5.X.X error
     3,528     3.7%   inactive-mailbox     messages rejected due to expired, inactive, or disabled recipient addresses, 5.X.X error
     2,206     2.3%   no-answer-from-host  messages bounces due to receiving no response from remote host after connecting, 4.X.X or 5.X.X error
     2,104     2.2%   policy-related       messages refused or blocked due to general policy reasons, 5.X.X error
     1,761     1.9%   message-expired      messages bounced due to not being delivered before the bounce-after, 4.X.X error
       877     0.9%   routing-errors       messages bounced due to mail routing issues for recipient domain, 5.X.X error
       715     0.8%   bad-domain           messages bounced due to invalid or non-existing domains, 5.X.X error
       228     0.2%   quota-issues         messages rejected due to mailbox quota issues, 4.X.X or 5.X.X error
```

This report list the top level bounce information. The scope of the output is based on the type of report specified, and the above example is done using no additional options.

Note that the report is based only on the messages that have already been handled by PowerMTA. If the report is generated while PowerMTA is still attempting to deliver some of the messages, these will not be included in the report.

11.4.2.3 Delivery Times

```
Delivery Times And Numbers (total queue time):

   Delivery times per percentage of recipients (successful deliveries only):
```

```
                              recipients  avg. time  max. time  %rcpt.
                              ----------  ---------  ---------  ------
                               1,601,910    0:00:11    0:00:39     50%
                               1,922,292    0:00:19    0:01:32     60%
                               2,242,674    0:00:37    0:03:34     70%
                               2,563,056    0:01:15    0:07:51     80%
                               2,883,438    0:03:38    1:30:09     90%
                               3,043,629    0:09:17    2:04:30     95%
                               3,075,667    0:10:31    2:12:13     96%
                               3,107,705    0:11:50    2:26:06     97%
                               3,139,743    0:13:35    3:27:07     98%
                               3,171,781    0:15:39    3:55:34     99%
                               3,203,820    0:18:33   16:11:17    100%


   Deliveries per time increment (successful deliveries only):

           interval (between)  histogram                recipients  %rcpt.
           ------------------  -----------------------  ----------  ------
            0 -  5 seconds     ***                         517,753   16.2%
            6 - 30 seconds     ******                      952,742   29.7%
           31 - 60 seconds     **                          328,600   10.3%
            1 -  2 minutes     *                           223,472    7.0%
            2 -  5 minutes     **                          320,070   10.0%
            5 - 10 minutes     **                          350,851   11.0%
           10 - 30 minutes                                 122,132    3.8%
           30 - 60 minutes                                  35,897    1.1%
            1 -  2 hours       *                           171,876    5.4%
            2 -  3 hours                                    97,408    3.0%
            3 -  4 hours                                    54,527    1.7%
            4 -  5 hours                                    18,589    0.6%
            5 -  6 hours                                     4,605    0.1%
            6 -  8 hours                                     3,560    0.1%
            8 - 10 hours                                       376    0.0%
           10 - 12 hours                                     1,016    0.0%
           after 12 hours                                      346    0.0%


   Cumulative deliveries per time increment (successful deliveries only):

           interval            histogram                recipients  %rcpt.
           ------------------  -----------------------  ----------  ------
           within 5 seconds    ***                         517,753   16.2%
           within 30 seconds   **********                1,470,495   45.9%
           within 1 minute     ************              1,799,095   56.2%
           within 2 minutes    ***************           2,022,567   63.1%
           within 5 minutes    ****************          2,342,637   73.1%
           within 10 minutes   *******************       2,693,488   84.1%
           within 30 minutes   ********************      2,815,620   87.9%
           within 1 hour       ********************      2,851,517   89.0%
           within 2 hours      *********************     3,023,393   94.4%
           within 3 hours      **********************    3,120,801   97.4%
           within 4 hours      **********************    3,175,328   99.1%
           within 5 hours      **********************    3,193,917   99.7%
           within 6 hours      **********************    3,198,522   99.8%
           within 8 hours      **********************    3,202,082   99.9%
           within 10 hours     ***********************   3,202,458  100.0%
           within 12 hours     ***********************   3,203,474  100.0%
           after 12 hours      ***********************   3,203,820  100.0%
```

This section and respective outputs are always generated and the scope of the outputs are based on the type of report specified, i.e., global or by originator/envelope ID. They also are primarily based on the difference between each message's queue timestamp and transfer timestamp for only the messages that were successfully delivered by PowerMTA.

As mentioned in Section 1.3.2.7, for the avg. time and max. time calculations, each recipient is counted by itself, even if a message is transferred in multi-recipient SMTP transactions.

293

The `avg. time` is calculated by adding together the fastest n% transfer times (transfer timestamp - queue timestamp) and then dividing this sum by the number of recipients that make up n% of the total deliveries. For example, in the 80% fastest delivery line in the top output, one would read that it took an average of 21 seconds for each message to each recipient to be delivered once each was queued, for the fastest 80% of all transfer times (or to 7,504,928 recipients).

The `max. time` is the longest transfer time (transfer timestamp - queue timestamp) for one message from all of the fastest transfer times for the number of recipients. In the 80% fastest delivery case, one would read that the longest it took for a message to be transferred, for the fastest 80% of all transfer times (or to 7,504,928 recipients), was 4 minutes and 24 seconds. Conversely, all of the 80% messages were transferred in equal or less time. Note that this does not show total campaign duration time since these figures are only based on each message's own transfer time.

It is important to see with this report the overall impact that slow or initially unreachable domains have on throughput efficiency, especially if it happens to be a domain with a large number of queued recipients.

11.4.2.4 Message Counts

```
Global Message Counts And Sizes:

   Message counts:

                                      recipients           volume
                                      ----------    ---------------
                          deliveries   3,203,820   36,046,885.7 kb
                          bounces         94,457
                          total        3,298,277   36,046,885.7 kb


   Message sizes:

                                   min. size  avg. size  max. size
                                   ---------  ---------  ---------
                                     2.1 kb    11.3 kb   122.4 kb
```

This report list the basic numbers associated on the mailings being reported. Totals for delivered, bounced, and size of messages are included.

11.4.2.5 Top Bounce

```
Top Bounce Domain Breakdowns:

         domain  recipients  total bounced  5.X.X bounces  4.X.X bounces
   -------------  ----------  -------------  -------------  -------------
       aol.com     202,123         26,906         26,906              0
  earthlink.net     35,028         12,148         12,148              0
   gms.aol.com     234,528          8,501          8,501              0
  bellsouth.net     28,401          7,647          7,647              0
       msn.com      10,138          7,636          7,636              0
     yahoo.com   1,548,313          4,197          4,197              0
     gmail.com       6,563          2,140          2,140              0
```

```
        hotmail.com    207,995           1,650          1,650               0
        comcast.net     67,001           1,429          1,428               1
        verizon.net     12,338           1,344            107           1,237

...

    bellsouth.net

               domain  recipients  total bounced  5.X.X bounces  4.X.X bounces
          -------------  ----------  -------------  -------------  -------------
          bellsouth.net      28,401          7,647          7,647              0

   bounced  %bounced  dsn code (reason)
   -------  --------  ----------------
     7,401     96.8%  5.3.2 (system not accepting messages); 550 Too many invalid recipients
       149      1.9%  5.0.0 (undefined status); 550 Invalid recipient: ...@...
        95      1.2%  5.0.0 (undefined status); 501 #5.1.1 bad address ...@...
         2      0.0%  5.0.0 (undefined status); 550 Too many invalid recipients

...
```

The top bounce domain breakdown shows detailed bounce reasons for the top domains.
By default, the top 10 domains (in terms of bounces) are printed, but this can be changed
with the -topDomains num option. Data was removed from the above output to save
space in this documents.

For each domain the top bounce reasons are given as absolute number of bounces and as
percentage of the bounces. The DSN (delivery status notification) code gives the reason
for that bounce, as received by the foreign mail server. As a trade-off between detail and
usability the DSN reason is only slightly modified (for example, email addresses in
bounce messages are also replaced by ...@...to be able to generalize the bounces into
bounce categories).

So, while the total bounces breakdown provides an aggregated overview of the bounces,
the top domains breakdown gives full detail on bounces grouped by domain.

11.4.2.6 Top Domains

```
Top Domains

   Domains with largest number of queued recipients:

                    domain  recipients  bounced  avg. time  max. time  %rcpt.
          ----------------------  ----------  -------  ---------  ---------  ------
                   aol.com      92,891      326    0:00:02    0:01:23   23.2%
               hotmail.com      48,848    1,028    0:00:12    0:27:59   12.2%
                 yahoo.com      34,355    1,912    0:00:25    0:28:42    8.6%
                 webtv.net      12,966      104    5:12:28    8:25:40    3.2%
               netzero.net       7,648        0    0:00:02    0:00:28    1.9%


   Domains with highest number of bounces:

                    domain  recipients  bounced  avg. time  max. time  %rcpt.
          ----------------------  ----------  -------  ---------  ---------  ------
                 yahoo.com      34,355    1,912    0:00:25    0:28:42    8.6%
               hotmail.com      48,848    1,028    0:00:12    0:27:59   12.2%
```

295

```
            earthlink.net      4,958       543   0:00:07   0:01:25   1.2%
         worldnet.att.net      3,002       353   0:00:01   0:00:13   0.7%
                juno.com       5,039       335   0:00:02   0:00:35   1.3%


  Domains with highest average delivery times:

                      domain  recipients  bounced  avg. time  max. time  %rcpt.
          ----------------------  ----------  -------  ---------  ---------  ------
                    paris.net           1        0   40:50:30   40:50:30   0.0%
                erpenbeck.com           2        0   27:57:24   47:13:33   0.0%
             hvrsd.k12.nj.us            1        0   21:42:25   21:42:25   0.0%
                  in.usda.gov           3        0   15:06:56   39:49:46   0.0%
                    acbl.org            2        0   12:53:25   17:35:36   0.0%
```

This report shows the counts and throughput for the top domains based on:

- the largest number of recipients initially submitted and queued for delivery,
- the highest number of bounces handled by PowerMTA, and
- the highest average delivery times.

The scope of the outputs are based on the type of report specified, i.e., global or by originator, envelope ID, or virtual mail transfer agent.

For all three sections, the `recipients` column shows the total number of recipients submitted and queued for delivery for the domains, while the `bounced` column provides just the number of recipients that PowerMTA itself bounced (local bounces). Note that this is not a count of all of the bounces since any bounces happening after PowerMTA has transferred the message to an intermediate mailer count as successful transfers. The difference between the recipients and bounced columns is then the number of recipients that were successfully transferred by PowerMTA.

For all three sections, the `avg. time` and `max. time` calculations are based on the difference between each message's queue timestamp and transfer timestamp. Only messages that were successfully delivered are included; bounces and other delivery reports are not. Messages transferred in multi-recipient (multi-"RCPT TO") SMTP transactions are broken down and each recipient is counted as a separate "delivery", each with its own transfer rate equal to that of the whole transaction itself. This is not an issue with single RCPT TO messages since messages delivered already equals recipients delivered.

The `avg. time` is calculated by adding together all of the transfer times (transfer timestamp - queue timestamp) to the respective domain for each recipient, and then dividing this by the number of recipients to the domain. Using AOL as an example in the first output, one would read that it took an average of 2 seconds for each message for each recipient to be delivered, once each message was queued, for the 92,891 recipients at AOL.com.

The `max. time` is the longest transfer time (transfer timestamp - queue timestamp) for one message to the respective domain. In the AOL case, one would read that the longest it took for one message to be delivered to AOL was 1 minute and 23 seconds. Conversely, all other messages to AOL for the 92,891 recipients were transferred in equal or less time.

The `%rcpt` column provides the percentage of recipients submitted and queued for the domain to the total number of recipients across the scope of the report. For example, AOL's 92,891 queued recipients were 23.2% of the 400,761 total queued recipients.

11.4.2.7 Top Rates

```
Top Rates

Top recipient delivery rates (including reports):

     measured over   rcpt./min.    recipients  beginning on
     ------------   ----------   ------------  -------------------
     one minute         19,896        19,896  2003-04-03 02:02:05
     five minutes       17,951        89,756  2003-04-03 05:10:27
     one hour           16,245       974,702  2003-04-03 04:28:34


Top throughput rates (including reports):

     measured over      kbyte/s         kbytes  beginning on
     ------------   ----------   ------------  -------------------
     one minute        5,517.4      331,045.8  2003-04-02 18:34:42
     five minutes      3,989.6    1,196,891.6  2003-04-03 05:35:50
     one hour          3,219.0   11,588,374.9  2003-04-03 05:00:56
```

This report shows the top throughput rates realized per recipient and the top data transfer rates during a time interval of one and five minutes and one hour. The scope of the output is *always* global and such is always based on the total numbers in the whole accounting file. All messages, normal deliveries as well as reports (including bounces) are taken into account when calculating these rates.

The top recipient delivery rates are calculated for each time increment by sorting all accounting records by delivery time and then searching for the time interval during which the largest number of recipients were handled. This number is then entered in then `recipients` column. The `rcpt./min.` column provides a uniform recipient transfer measurement across the three time intervals, and is calculated by taking the numbers in the `recipients` column and dividing each by the number of minutes that make up each respective time interval (i.e. dividing the five minutes result by five and the one hour result by sixty). The `messages` column shows the corresponding number of messages that were actually sent (i.e., number of SMTP transactions) while handling these recipients. This information is important when comparing performance numbers, since delivering a single message to multiple recipients is a lot less resource intensive than delivering individual messages.

The top throughput rates are calculated in the same manner as the recipient delivery rates, however the top amount of data transferred per time interval is reported rather than the number of recipients. This transfer rate output is independent to the recipient delivery rate output, so they may or may not reflect the same messages. The `kbytes/s` column provides a uniform data transfer measurement across the three time increments, and is calculated by taking the numbers in the `kbytes` column and dividing each by the number of seconds that make up each respective time increment (i.e., dividing the five minutes result by 300 and the one hour result by 3600). The `messages` column shows the corresponding number of messages that were actually sent comprising the number of kilobytes shown.

11.4.2.8 VMTA Summary

```
VirtualMTA Traffic Summary:

  VirtualMTA            IP address        Recipients  Delivered  Bounced  Bounced%
  -------------------   --------------    ----------  ---------  -------  --------
  vmta1                 192.168.0.97       489,134    489,134         0     0.0%
  vmta2                 192.168.0.99       466,068    466,068         0     0.0%
  vmta3                 192.168.0.98       461,980    461,980         0     0.0%
  vmta4                 192.168.0.132      240,937    240,937         0     0.0%
  vmta5                 192.168.0.134      195,063    195,063         0     0.0%
  vmta6                 192.168.0.136      194,362    194,362         0     0.0%
  vmta7                 192.168.0.135      192,084    192,084         0     0.0%
  vmta8                 192.168.0.133      185,040    185,040         0     0.0%
  vmta9                 192.168.0.87       123,175    123,175         0     0.0%
  vmta10                192.168.0.88       119,081    119,081         0     0.0%
  vmta11                192.168.0.86       111,882    111,882         0     0.0%
  vmta12                0.0.0.0             56,713          0    56,713   100.0%
  vmta13                192.168.0.171       45,823     45,823         0     0.0%
  vmta14                192.168.0.172       45,554     45,554         0     0.0%
  vmta15                192.168.0.168       45,461     45,461         0     0.0%
  vmta16                192.168.0.169       45,186     45,186         0     0.0%
  vmta17                192.168.0.170       44,714     44,714         0     0.0%
  vmta18                192.168.0.163       29,997     29,997         0     0.0%
  vmta19                192.168.0.158       29,201     29,201         0     0.0%
  vmta20                192.168.0.161       28,767     28,767         0     0.0%
  vmta21                192.168.0.164       28,287     28,287         0     0.0%
  vmta22                192.168.0.157       27,652     27,652         0     0.0%
  vmta23                192.168.0.162       27,377     27,377         0     0.0%
  vmta24                0.0.0.0             27,104          0    27,104   100.0%
  vmta25                0.0.0.0              8,526          0     8,526   100.0%
  vmta26                192.168.0.126        3,634      3,634         0     0.0%
  vmta27                192.168.0.122        3,348      3,348         0     0.0%
  vmta28                192.168.0.124        3,250      3,250         0     0.0%
  vmta29                192.168.0.125        3,247      3,247         0     0.0%
  vmta30                192.168.0.121        3,187      3,187         0     0.0%
  vmta31                192.168.0.123        3,128      3,128         0     0.0%
  vmta32                192.168.0.200        1,872      1,872         0     0.0%
  vmta33                192.168.0.174        1,693      1,693         0     0.0%
  vmta34                192.168.0.173        1,680      1,680         0     0.0%
  vmta35                192.168.0.175        1,590      1,590         0     0.0%
  vmta36                0.0.0.0                943          0       943   100.0%
  vmta37                0.0.0.0                782          0       782   100.0%
  vmta38                0.0.0.0                387          0       387   100.0%
  (default)             192.168.0.238          267        267         0     0.0%
  vmta39                192.168.0.185           33         33         0     0.0%
  vmta40                192.168.0.184           27         27         0     0.0%
  vmta41                192.168.0.186           27         27         0     0.0%
  vmta42                192.168.0.181            5          5         0     0.0%
  vmta43                192.168.0.180            4          4         0     0.0%
```

```
vmta44              192.168.0.182        3         3        0      0.0%
(default)           0.0.0.0              2         0        2    100.0%

Total                            3,298,277  3,203,820   94,457     2.9%
```

VMTA Summary shows high level counters on the amount of messages that were sent to individual VirtualMTAs. This information includes a breakdown of deliveries, bounces, and percentage of bounces per VirtualMTA. The delivering IP is also included to help in determining if there is a reputation/sending issue with a given address.

11.4.2.9 VMTA Time Breakdown

```
VirtualMTA Traffic Breakdown By Time:

  VirtualMTA: vmta1, IP Address: 192.168.0.97

  Daily Breakdown:

  Date        Recipients  Delivered  Bounced  Bounced%
  ----------  ----------  ---------  -------  --------
  2007-01-16    416,028    416,028        0      0.0%
  2007-01-17     73,106     73,106        0      0.0%

  Hourly Breakdown:

  --------------------- 2007-01-16 ---------------------
  Interval        Recipients  Delivered  Bounced  Bounced%
  -------------   ----------  ---------  -------  --------
  00:00 - 00:59           0          0        0      0.0%
  01:00 - 01:59           0          0        0      0.0%
  02:00 - 02:59           0          0        0      0.0%
  03:00 - 03:59           0          0        0      0.0%
  04:00 - 04:59           1          1        0      0.0%
  05:00 - 05:59         207        207        0      0.0%
  06:00 - 06:59           1          1        0      0.0%
  07:00 - 07:59           2          2        0      0.0%
  08:00 - 08:59           2          2        0      0.0%
  09:00 - 09:59           0          0        0      0.0%
  10:00 - 10:59          40         40        0      0.0%
  11:00 - 11:59          35         35        0      0.0%
  12:00 - 12:59          80         80        0      0.0%
  13:00 - 13:59         140        140        0      0.0%
  14:00 - 14:59          35         35        0      0.0%
  15:00 - 15:59           0          0        0      0.0%
  16:00 - 16:59           0          0        0      0.0%
  17:00 - 17:59           3          3        0      0.0%
  18:00 - 18:59      24,712     24,712        0      0.0%
  19:00 - 19:59      99,934     99,934        0      0.0%
  20:00 - 20:59      92,585     92,585        0      0.0%
  21:00 - 21:59      74,124     74,124        0      0.0%
  22:00 - 22:59      68,374     68,374        0      0.0%
  23:00 - 23:59      55,753     55,753        0      0.0%

  --------------------- 2007-01-17 ---------------------
  Interval        Recipients  Delivered  Bounced  Bounced%
  -------------   ----------  ---------  -------  --------
  00:00 - 00:59      67,536     67,536        0      0.0%
  01:00 - 01:59       5,460      5,460        0      0.0%
  02:00 - 02:59         104        104        0      0.0%
  03:00 - 03:59           6          6        0      0.0%
  04:00 - 04:59           0          0        0      0.0%
  05:00 - 05:59           0          0        0      0.0%
  06:00 - 06:59           0          0        0      0.0%
```

299

```
07:00 - 07:59            0            0            0        0.0%
08:00 - 08:59            0            0            0        0.0%
09:00 - 09:59            0            0            0        0.0%
10:00 - 10:59            0            0            0        0.0%
11:00 - 11:59            0            0            0        0.0%
12:00 - 12:59            0            0            0        0.0%
13:00 - 13:59            0            0            0        0.0%
14:00 - 14:59            0            0            0        0.0%
15:00 - 15:59            0            0            0        0.0%
16:00 - 16:59            0            0            0        0.0%
17:00 - 17:59            0            0            0        0.0%
18:00 - 18:59            0            0            0        0.0%
19:00 - 19:59            0            0            0        0.0%
20:00 - 20:59            0            0            0        0.0%
21:00 - 21:59            0            0            0        0.0%
22:00 - 22:59            0            0            0        0.0%
23:00 - 23:59            0            0            0        0.0%
```

The VMTA time breakdown shows bounce counts for the top VirtualMTAs by time interval. By default all the VirtualMTAs are printed. For each VirtualMTA the bounce counts are given as absolute number of bounces and as percentage of the bounces for a given time period.

11.4.2.10 VMTA Top Bounces

```
VirtualMTA Top Bounce Domains:

   VirtualMTA: vmta1, IP Address: 192.168.0.97

   Domain          Recipients  Delivered  Bounced  Bounced%
   ---------------  ----------  ---------  -------  --------
   yahoo.com          387,535    387,535        0      0.0%
   verizon.net         18,710     18,710        0      0.0%
   aol.com             17,314     17,314        0      0.0%
   sbcglobal.net       14,241     14,241        0      0.0%
   juno.com             5,200      5,200        0      0.0%
   netzero.net          3,186      3,186        0      0.0%
   cox.net              2,300      2,300        0      0.0%
   excite.com             957        957        0      0.0%
   bellsouth.net          705        705        0      0.0%
   netscape.com           594        594        0      0.0%
```

The VMTA Top Bounces report shows bounce counts for all the VirtualMTAs. By default, the top 20 domains (in terms of bounces) are printed, but this can be changed with the --topBounced <num> option.

For each domain the top bounce reasons are given as absolute number of bounces and as percentage of the bounces. So, while the top bounces report provides an aggregated overview of the bounces, the VMTA top bounces report gives high level data on bounces grouped by VirtualMTA.

300

11.5 The `acctfind` Accounting Reporting Application

`acctfind` is a powerful accounting file search and reporting tool. It can be found in the `\pmta\bin` directory on Windows, and in the `/usr/sbin` directory (named `pmtaacctfind`) on Unix.

The tool is not required for exporting data from the accounting file because the file is no longer in a proprietary binary format. With the CSV accounting file written by PowerMTA, reporting can now be done without this tool. This tool, however, will still work with the CSV file format, and is provided to assist with extracting subsets of the data.

`acctfind` allows one to search for records in the accounting file, as well as to produce a custom output based on specific fields in the matching records. For example, one can easily use `acctfind` to generate a list of all addresses that bounced with a persistent (5xx) error for a particular campaign, sorted by the bounce reason, in a tab separated output that contains the recipient address first, followed by the MAIL FROM/originator address, then the bounce code and reason.

`acctfind` currently supports seven different output formats including standard, comma separated values, tab separated values, html, xml, "pretty" xml, and custom-defined outputs.

This Section provides a quick overview of the feature set and options, along with sample commands, however the more detailed help obtained by passing the `--help` option to `acctfind` itself should be referenced for further insight.

The general format for the command is

```
acctfind [options] accounting files ...
```

The top level segmentation options are based on the record types, which include searching for matches in records for messages successfully delivered (`--delivered`), bounce reports delivered (`--bounced`), run status records (`--status`), and general mta information (`--mta`).

For example,

```
acctfind --bounced acct.dat
```

would search for matching records only among the bounce report records. If none of these are specified in the command, the default is to search for matches in both message delivered records and bounce report records.

All of the fields in the accounting records (listed in [Section 1.1.1](#) & [Section 1.1.2](#)) are available for matching (with the `--match` option), and the match pattern can be either a perl regular expression (e.g., `m/^5\./`) or some string for case-independent string matching.

If you want to restrict the matching to some fields, you can do that by providing a list of fields to match enclosed in square brackets in front of the pattern (e.g. `--match [from,to]port25.com`).

Shortcut options for matching single fields are also available. For example, `--match [to]port25` is equivalent to `--to port25`. To further refine the output, more than one matching needle can be specified. For example, to get a listing of all `yahoo.com` addresses that bounced with a persistent (5xx) error, you would have the following option combinations in the command

```
acctfind --bounced --dsnStatus m/^5\./ --to yahoo.com acct.dat
```

As mentioned, seven different output formats are supported. You choose the desired format using the `--output` option, along with the format and the fields desired to be outputted. For example,

```
acctfind --delivered --to hotmail.com --output csv[to,from,envid,vmta,dstIp] acct.dat
```

would search all message delivered records for recipient addresses at `hotmail.com` and generate a comma separated output which lists, on each line,

- the hotmail recipient address,
- the message originator address,
- the envelope ID used for tracking the message,
- the VirtualMTA used to deliver the message, and
- the IP address of the Hotmail machine that accepted the message.

To get an HTML output vs. csv output of the same, you would run

```
acctfind --delivered --to hotmail.com --output html[to,from,envid,vmta,dstIp] acct.dat
```

A full detailed listing and description of all of the options available can be found in the help by running `acctfind --help`. Included below are a few sample commands and options to help better understand the formats and capabilities. Do not hesitate to contact us at `support@port25.com` with any questions or when seeking the proper command sequence for a desired result.

11.5.1 Examples

1. To generate a list of all addressess that bounced in a comma separated output, that contains the recipient address first, followed by the MAIL FROM/originator address, then VirtualMTA used, then the bounce codes and reason, sorted by the bounce reason (DSN Diagnostics):

   ```
   acctfind --bounced --sortBy dsnDiag --output tsv[to,from,vmta,dsnStatus,dsnDiag] acct.dat
   ```

   Note that the --sortBy option is only supported on Unix at this time.

2. To generate a list of all addresses that bounced with a persistent (5xx) error, in a comma separated output that contains the recipient address first, followed by the MAIL FROM/originator address, envelope ID, then the bounce code and reason:

   ```
   acctfind --bounced --dsnStatus m/^5\./ --output csv[to,from,envId,dsnStatus,dsnDiag] acct
   ```

3. To generate a list of all hotmail.com addresses that bounced, in an HTML output that contains the recipient address, and bounce reasons:

   ```
   acctfind --bounced --to hotmail --output html[to,dsnStatus,dsnDiag] acct
   ```

4. To generate a list of all hotmail.com addresses that were successfully transferred, in a "pretty XML" output that contains the recipient addresses, VirtualMTA used to deliver the messages, the queue and transfer time stamps in ISO format, and the IP address of the hotmail machine that accepted the messages:

   ```
   acctfind --iso-times --delivered --to hotmail --output
   prettyxml[to,vmta,timeQueued,timeDelivered,dstIp] acct.dat
   ```

5. To search for a particular e-mail address (test@port25.com) to see whether it was delivered or bounced, with a tsv output showing the address, whether the message failed, and the ISO-formatted queue time and delivery time:

   ```
   acctfind --iso-times --to test@port25.com --output tsv[to,dsnAction,timeQueued,timeDelivered] acct.dat
   ```

6. To search for a particular e-mail address (test@port25.com) to see whether it was delivered or bounced, with the standard output:

   ```
   acctfind --iso-times --to support acct.dat
   ```

7. To search for a particular header that is being logged via the "`acct-log-header`" directive:

```
acctfind --output csv[to,from,header:x-foobar] acct.dat
```

11.6 Bounce Categories

For easier classification of the actual reason for a bounce, `pmtastats` groups bounces in *bounce categories*. All bounces in a category have probably the same root cause, e.g. all messages in a category may be considered spam or another category may indicate the recipient is unknown. Using the various bounce codes and DSN reasons, `pmtastats` aggregates these message in likely bounce categories. This gives a good overview of why messages could not be delivered.

The categories are customizable using the <bounce-category> tag (see Section 3.2). Any new or custom categories would be show in the pmtastats output as well as in the accounting file itself.

The bounce category is included in the accounting file (the `bounceCat` tag, see Section 1.1.1). It can be obtained directly from the CSV file or be used in `acctfind`.

Currently, these bounce categories are defined:

`bad-configuration`	messages rejected due to configuration issues with remote host, 5.X.X error
`bad-connection`	messages bounced due to bad connection issues with remote host, 4.X.X error
`bad-domain`	messages bounced due to invalid or non-existing domains, 5.X.X error
`bad-mailbox`	messages rejected due to bad, invalid, or non-existent recipient addresses, 5.X.X error
`content-related`	messages refused or blocked due to content related reasons, 5.X.X error
`inactive-mailbox`	messages rejected due to expired, inactive, or disabled recipient addresses, 5.X.X error
`invalid-sender`	messages bounced due to invalid DNS or MX entry for sending domain
`message-expired`	messages bounced due to not being delivered before the bounce-after, 4.X.X error

no-answer-from-host	messages bounces due to receiving no response from remote host after connecting, 4.X.X or 5.X.X error
other	messages rejected due to other reasons, 4.X.X or 5.X.X error
policy-related	messages refused or blocked due to general policy reasons, 5.X.X error
protocol-errors	messages rejected due to SMTP protocol syntax or sequence errors, 5.X.X error
quota-issues	messages rejected or blocked due to mailbox quota issues, 4.X.X or 5.X.X error
relaying-issues	messages refused or blocked due to remote mail server relaying issues, 5.X.X error
routing-errors	messages bounced due to mail routing issues for recipient domain, 5.X.X error
spam-related	messages refused or blocked due to spam related reasons, 5.X.X error
virus-related	messages refused or blocked due to virus related reasons, 5.X.X error

Note that this list and the classifications may be subject to change, since DSN status messages may vary.

11.7 Libraries Available for Parsing CSV Records

Each API example program shipped with the APIs also has a comment in the beginning of the file pointing to the CSV library being used.

In API examples we use:

- JavaCSV (http://sourceforge.net/projects/javacsv/) is a free, easy to use Java CSV library. It can read the headers from the first line and then later in the program you can access individual fields by their name. (Or by number, if you wish.)
- Lumenworks CsvReader (http://www.codeproject.com/cs/database/CsvReader.asp) is a free library for .NET (you have to register with the codeproject website, though). We use it for the .NET API examples. It is similar to JavaCSV.
- CsvJDBC (http://csvjdbc.sourceforge.net) can be used like a JDBC driver that is based on a CSV file instead of a real database. It is not recommended for reading the CSV files, unless the programmer is very much used to JDBC and doesn't want to learn a new library. There are probably similar Microsoft classes in the ODBC area that do the same thing.

305

- With Text::CSV_XS you create a csv object that you can then ask to read and parse a line from a data source you provide (writing is also possible but not needed since the accounting files are only read). The easiest way to provide such a data source is to use the IO::Handle module that comes with Perl that allows you to pass an I/O handle as a data source.

With the first call to the getline() method of the csv object you get an array with the column headers, with the subsequent calls you get an array with the data for one record in the same order as the column headers (this is because the first line in an accounting field contains the column headers, it's no special treatment from the getline() method). So access to the data is only possible by column index. If you want access by column headers the accountingByIp.pl and splitter.pl examples contain a sub that converts the array to a hash with the column headers as keys.

Beware of libraries that attempted to buffer (i.e., keep in memory) the whole CSV input. With the expected size of the accounting files, it would be strongly adviseable to not use such a memory intensive parser.

We recommend against writing one's own CSV reader, as the CSV format can be more difficult to work with than it initially looks.

12. Processing Inbound email

12.1 Introduction

There may be times when PowerMTA needs to handle inbound mail outside of messages created by a submission (e.g. list manager, CRM, etc.) application. These messages may be remote (asynchronous) bounces, corporate mail, or any other type of email originating from outside an organization's infrastructure. In these cases, PowerMTA can accept the messages and do one of a few things with the messages. It can:

- Relay them to another mail server
- Deliver them to file on the local system
- Discard them (blackhole)
- Deliver to pipe (local mail processing application)
- Process asynchronous (remote) bounces
- Process feedback loop emails

Any combination of these options may be used, and PowerMTA will create a subsequent delivery record in the accounting file for mail that is processed in one of these manners. Remote bounce and feedback loop parsing coming soon. Contact Port25 Solutions for more information.

12.2 Relaying

To relay messages to another mail server, PowerMTA will need to be configured to accept the messages, and then deliver them to their final destination.

```
relay-domain yourdomain.com
relay-domain otherdomain.com
relay-address bounce@bounces.yourdomain.com

<domain bounces.yourdomain.com>
  route [1.2.3.4]:25
</domain>

<domain otherdomain.com>
  route [5.6.7.8]:2525
</domain>
```

In the above example, messages for bounces.yourdomain.com are accepted for delivery and then forwarded to server 1.2.3.4 on port 25 and messages for otherdomain.com are forwarded to 5.6.7.8 on port 2525 and marked as such in the accounting file.

12.3 Deliver to File

To deliver messages to file, PowerMTA will need to be configured to accept the messages, and then deliver them to somewhere on the local disk system.

```
relay-domain bounces.yourdomain.com
relay-domain otherdomain.com
relay-address bounce@bounces.yourdomain.com

<domain bounces.yourdomain.com>
  type file
  file-format append-mbox
  file-destination /etc/pmta/inbound/$domain
</domain>

<domain otherdomain.com>
  type file
  file-format newfile-pickup
  file-destination /etc/pmta/inbound/$domain
</domain>
```

In the above example, messages for bounces.yourdomain.com and otherdomain.com are accepted for delivery and then each message is delivered to the individual files (based on domain) on the local disk and marked as such in the accounting file.

12.4 Discarding

To discard (blackhole) the inbound messages, PowerMTA will need to be configured to accept the messages, and then discard them based on domain.

```
relay-domain bounces.yourdomain.com
relay-domain otherdomain.com
relay-address bounce@bounces.yourdomain.com

<domain bounces.yourdomain.com>
  type discard
</domain>

<domain otherdomain.com>
  type discard
</domain>
```

In the above example, messages for bounces.yourdomain.com and otherdomain.com are accepted for delivery, then discarded and marked as such in the accounting file.

12.5 Delivering to local application (pipe)

To deliver messages to a local application via pipe, PowerMTA will need to be configured to accept the messages, and then deliver them based on domain.

```
relay-domain bounces.yourdomain.com
relay-domain otherdomain.com
relay-address bounce@bounces.yourdomain.com

<domain bounces.yourdomain.com>
  type pipe
  command "/my/bounce/processor --envid \"$envid\" \"$user\""
</domain>

<domain otherdomain.com>
  type pipe
  command "/my/bounce/processor --envid \"$envid\" \"$user\""
</domain>
```

In the above example, messages for bounces.yourdomain.com and otherdomain.com are accepted for delivery, then piped to the custom application (with associated parameters) on the local file system and marked as such in the accounting file.

12.6 Processing Asynchronous (Remote) Bounces and remote status reports

Remote bounce processing works as a "filter" located after receipt of an email and before it is routed to a delivery queue. If PowerMTA recognizes the format and no errors occur while processing it, it writes a new type of accounting record ("rb" and "rs", see below). For those emails you have the option of ending the email there or letting it pass to regular delivery. If the message isn't recognized or an error occurs, it is passed to a delivery queue and delivered normally (by whatever method assigned to the domain -- usually 'file' or 'pipe').

As a note, the same domain and/or address for <address-list> can be used for both remote bounces and feedback loops. If this is done, PowerMTA will check the message first for a remote bounce, then if unmatched, for a feedback loop email.

The email address used should be unique for each email sent to allow for more detailed tracking. For example, instead of:
  bounce@bounce.yourdomain.com

Use of postmaster@ or abuse@ will not work.

The following should be considered instead:
  customer1-campiagn563-uniquieID498294@bounce.yourdomain.com

Accounting Records:

There are two kinds of records for asynchronous bounces: "rb" for remote bounce and "rs" for reporting other kinds of remote status (used for DSN positive and "delayed"

309

status notifications). Most people will only be interested in "rb" records. The following columns are available for these records:

Name	Description
timeLogged	current time
orig	originator (taken from report's recipient address)
rcpt	(final) recipient
orcpt	original recipient
dsnAction	DSN action
dsnStatus	DSN status
dsnDiag	DSN diagnostic
dsnMta	DSN remote MTA
dsnReportingMta	DSN reporting MTA
bounceCat	bounce category
srcType	type of source from which the report was received
srcMta	MTA from which the report was received
envId	envelope ID
header_<name>	header extracted from original email

Configuration:

You define a list of email addresses that are bounce addresses, i.e., original MAIL FROM addresses. Only email received for those addresses gets filtered/looked at looking for bounces. Use of the following requires a restart of the PowerMTA service. You can define entire domains which get filtered, or for more granular control you can use regular expressions to match entire addresses with the option to deliver recognized emails is configured as:

```
<bounce-processor>
        deliver-unmatched-email yes         # default: no
        deliver-matched-email yes           # default: no
        forward-unmatched-to auto-feedback@port25.com
        forward-errors-to auto-feedback@port25.com
    <address-list>
        domain domain.to.filter             # whole domain
        address /regex@domain.to.filter/    # regex
    </address-list>
</bounce-processor>

#
# Required for delivery of emails regardless of
# deliver-unmatched-emails being set to yes or no
#
<domain domain.to.filter>
  route [1.2.3.4]:25
</domain>
```

310

The accounting records are configured just like the other types in the configuration. The example below has both local (sync) and remote (async) bounces written to the same file, and includes a custom "x-id" header:

```
<acct-file /var/log/pmta/bounces.csv>
    records b, rb
    record-fields b  *,header_x-id
    record-fields rb *,header_x-id
</acct-file>

<acct-file /var/log/pmta/status.csv>
    records rs
    record-fields rs *,header_x-id
</acct-file>
```

After intial configuration a restart of the PowerMTA service is required for asynchronous bounce processing to work.
If you are having trouble, and need to turn on logging, setting the following global directive:

```
<domain {bounce}>
  log-messages yes
</domain>
```

12.7 Processing Feedback Loop Emails

To enable feedback loop processing, you add something like:

```
<feedback-loop-processor>
        deliver-unmatched-email no
        deliver-matched-email yes              # default: no
        forward-unmatched-to auto-feedback@port25.com
        forward-errors-to auto-feedback@port25.com
         <address-list>
             address /fbl@fbl.yourdomain.com/
        </address-list>
</feedback-loop-processor>

#
# Required for delivery of emails regardless of
# deliver-unmatched-emails being set to yes or no
#
<domain mydomain.com>
  route [1.2.3.4]:25
</domain>
```

to the configuration file. The "forward-..." directives can be specified more than once to have email forward to multiple addresses. "<address-list>" accepts individual domains

311

like in "domain foo.com" and regular expressions like in the example above, so you have great flexibility in specifying the addresses for which we should attempt FBL processing. As with bounce processing, you have the option to continue to deliver the emails after processing (e.g., to save them all to a folder) or to have them end there. Unmatched or failed emails always proceed to delivery.

As a note, the same domain and/or address for <address-list> can be used for both remote bounces and feedback loops. If this is done, PowerMTA will check the message first for a remote bounce, then if unmatched, for a feedback loop email.

The email address used should be something easy to parse. For example:
  fbl@fbl.yourdomain.com
PowerMTA should be the MX to handle these messages. If there are MTAs that handle the messages prior to being delivered to PowerMTA, PowerMTA may be unable to recognize the message as a FBL email.

Use of postmaster@ or abuse@ will not work.

To allow for more detailed tracking a custom header may be added to the email (assuming the remote mailer doesn't redact the header).
x-tracker: customer1-campiagn563-uniquieID498294

Custom headers are not included by default in the PowerMTA accounting file, so if the above header is needed in the accounting make sure to add it as a custom header.

Emails matching the address list go into a special {feedbackLoop} queue, where they are processed. You need also configure an accounting file to receive the resulting data:

```
<acct-file /var/log/pmta/fbl.csv>
    records feedback-loop
    record-fields f *,header_subject
</acct-file>
```

After intial configuration a restart of the PowerMTA service is required for feedback loop processing to work. "f" and "feedback-loop" can be used for the record name interchangeably. In the accounting file you actually see 'f'. The available columns (all of which, except for 'header', are included by default) are:

The following columns are available for f records:

Name	Description
timeLogged	current time
format	"arf" or "jmrp"
userAgent	name & version of the program that generated the report
envId	envelope ID

orig	originator (taken from report's recipient address)
rcpt	(final) recipient
reportingMta	the name of the MTA generating this feedback report
sourceIp	address of MTA from which the message was received
header_\<name\>	header extracted from original email
arf_\<name\>	field from ARF feedback-report

'header_From', 'header_Return-Path', 'header_X-job' and 'header_Subject' are included by default.

If a feedback report includes multiple recipients, a separate 'f' record is written for each recipient, repeating any common fields. Since a field requested via 'arf_...' may occur multiple times in the report, the column may contain a (CSV-encoded) list of values.

12.7.1 Hotmail Feedback Loop Emails

Hotmail does not follow the standard ARF format as the others. To get FBL emails to work for Hotmail messages, sign up for the RFC attachments option, since their "original" option does not include the key headers that PowerMTA parses on to indicate that it is an FBL, and the original format includes less info in general.

Because the RFC attachment option is still non-standard, the header fields from the original message will need to be extracted in contrast to standard ARF messages. Otherwise the RCPT field is empty by default in Hotmail JMRP accounting records.

In this case, PowerMTA will need to add the header "X-HmXmrOriginalRecipient" into the csv accounting file, which is where Hotmail includes the original recipient address that complained in the original message. To add custom headers to an acct record, add the header within the syntax "header_xxx" in the record-fields directive, for example:

```
<acct-file /var/log/pmta/fbl.csv>
  records feedback-loop
  record-fields f *,header_X-HmXmrOriginalRecipient
</acct-file>
```

Alternatively, the map-header-to-field feature can be used the custom header "X-HmXmrOriginalRecipient" to be be mapped to standard fields that already exist. With this, the same field for all FBL emails will be used. To map this Hotmail specific header to the existing rcpt field use something like the following:

```
<acct-file /var/log/pmta/fbl.csv>
  records feedback-loop
  record-fields f *
  map-header-to-field fbl header_X-HmXmrOriginalRecipient rcpt
```

313

</acct-file>

Lastly, what many sites do instead is create a custom header in each message that obfuscates the recipient address, perhaps using a mailing or campaign ID as well, for example:

x-rid: 12345-hdhgjkeuydfpdfldfd

and then have this header included in the FBL accounting file. When parsing the csv file just key on this header, since again, it is included in all FBLs sent back, which is the only thing consistent across the FBL generating sites.

12.8 Tracking a campaign in PowerMTA with a JobID

To track campaigns in PowerMTA it is best to use an x-job header. This will show as a JobID or a Job in PowerMTA and PowerMTA Management Console. For this to work properly the application feeding PowerMTA needs to set an x-job header. The value of this header should be unique to each campaign. For example:

```
x-job: UniqueID-Date
```

With this set, PowerMTA needs to know to look for the header with a setting similar to the following:

```
<source 0/0>
  process-x-job yes
</source>
```

If needed, an alternate header may be used with something similar to the following:

```
<source 0/0>
  jobid-header x-custom-header
  process-x-job yes
</source>
```

In the above example x-custom-header will be used as the JobID in place of x-job.

Once the above is set, PowerMTA will log the JobID to the accounting file once a delivery or bounce has been taken place. Information about messages still in the queue can be found on the Jobs tab of the PowerMTA web monitor or by using the "pmta show jobs" command.

When using a Job ID, it may become necessary during the course of a mailing to pause the job. Instances of when this would be needed might be to make sure the injected job was setup correctly, or that it is using the correct VirtualMTAs. The command to pause a job would be:

```
pmta pause job Campaign1234
```

And to resume the job:

```
pmta resume job Campaign1234
```

In the above example, Campaign1234 would be the Job ID that needs to be paused.

Lastly, additional information about current queue and historical reporting can be found in the PowerMTA Management Console dashboard, monitoring, and reporting sections.

12.9 Aliases / Forwarding

There may be a need to use aliases or to forward emails. This directive allows a list of email addresses to be defined for which PowerMTA should forward messages to different addresses. For example, if a message is received by PowerMTA for example@domain1abc.com, then forward the message on to support@port25.com.

Defining just the local part of the email address (the text to the left of the @) is also supported, however forwarding will only occur on addresses that have the matching local part and the domain of one of the local host names. Domain matching or regex pattern matching is not supported at this time.

The directives are applied on a global basis. The format is an email address (or local part of an address) followed by one or more than one comma separated email addresses. For example:

```
host-name yourdomain.com
host-name yourotherdomain.com
relay-domain example.com
relay-address user@example.com
<aliases>
  alias example@yourdomain.com support@port25.com,info@port25.com
  alias jdoe someone@example.com
</aliases>
```

In the above example emails sent to `example@yourdomain.com` will be forwarded to `support@port25.com` and `info@port25.com`. Emails sent to `jdoe@yourdomain.com` or `jdoe@yourotherdomain.com` will be forwarded to someone@example.com.

Multiple addresses can be defined with the same exact local part across different

domains. For example:

```
<aliases>
  alias newsletters@domain1abc.com support@port25.com
  alias newsletters@domain2def.com info@port25.com
  alias newsletters@domain3xyz.com support@port25.com,info@port25.com
</aliases>
```

Creating a file with more than 1000 addresses is not recommended at this time.
To forward all messages from a domain to a given address the following may be used:

```
<aliases>
  alias *@domain1abc.com support@port25.com
</aliases>
```

A. Deprecated APIs

A.1. Submission API

The C submission API consists of a header file (`submit.h`) as well as code included in a library file. By including the supplied header file and using the fuctions provided in the library, C/C++ programs can write directly into PowerMTA's spool.

A.1.1. Using the C Submission API

To send a message through the C submission API, you

1. Obtain a submission context by calling `SUBMIT_NewContext`;
2. Start a new message submission with `SUBMIT_StartMessage`;
3. Set the message's originator with `SUBMIT_SetOriginator` and specify the recipients by calling `SUBMIT_AddRecipient` for each of them;
4. Optionally call `SUBMIT_AddHeader` to specify custom message headers;
5. Use a combination of `SUBMIT_AddContentLine`, `SUBMIT_AddContentLineLen` and `SUBMIT_AddContentBlock` to provide the message body;
6. Finish the message, completing the submission, with `SUBMIT_FinishMessage`.
7. When done submitting messages, discard the submission context using `SUBMIT_DeleteContext`.

Upon the first usage of `SUBMIT_AddContentLine`, `SUBMIT_AddContentLineLen` or `SUBMIT_AddContentBlock`, the API automatically supplies any missing headers and adds the empty line separating the message headers from the body. Header completion is performed as follows:

- If the `From:` header is missing (i.e., it has not been added using `SUBMIT_AddHeader`), it is automatically added based on the message's originator (specified with `SUBMIT_SetOriginator`);
- If the `Date:` header is missing, it is added.

Note that you can (and *should*, for best performance) continue to use the submission context to submit more messages after the first submission is completed. If your program is multithreaded, each thread should obtain and use its own submission context.

A.1.2. Function Reference

SUBMIT_NewContext

Creates a new submission context

```
SUBMIT_CTX* SUBMIT_NewContext()
```

Description:

Creates a new submission context. Zero is returned if not enough memory can be allocated.

SUBMIT_ResetContext

Resets the submission context

```
void SUBMIT_ResetContext(SUBMIT_CTX* ctx)
```

Arguments:

`ctx`

    Submission context

Description:

Resets the specified submission context, discarding the message currently being submitted, if not yet finished.

SUBMIT_DeleteContext

Deletes the submission context

```
void SUBMIT_DeleteContext(SUBMIT_CTX* ctx)
```

Arguments:

`ctx`

    Submission context

Description:

Deletes the specified submission context, reclaiming memory and other resources. If the message currently being submitted has not yet been finished, it is discarded.

SUBMIT_StartMessage

Starts the submission of a new message

```
BOOL SUBMIT_StartMessage(SUBMIT_CTX* ctx, const char* envelopeId,
    unsigned flags)
```

Arguments:

```
ctx
```
      Submission context
```
envelopeId
```
      Envelope ID for the message.
```
flags
```
      A combination of:

`SUBMIT_MSG_RETURN_HEADERS`	Specifies that any reports (e.g., bounces) should include only the message's headers.
`SUBMIT_MSG_RETURN_FULL`	Specifies that any reports (e.g., bounces) should include the full messsage.
`SUBMIT_MSG_BODY_7BIT`	Specifies that the message being submitted only includes 7-bit data.
`SUBMIT_MSG_BODY_8BIT`	Specifies that the message being submitted contains 8-bit (but not binary) data.
`SUBMIT_MSG_BODY_BINARY`	Specifies that the message being submitted has binary contents. Since binary contents are not yet currently supported, this option is a placeholder only. To provide binary content, you must first encode it (e.g., using base64 encoding).
`SUBMIT_MSG_OPT_VERP`	Enables the VERP (Variable Envelope Return Path) extension. This non-standard SMTP extension is described in http://www.port25.com/support/rfcs/draft-varshavchik-verp-smtpext-03.txt.

Description:

Starts the submission of a new message. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

The RETURN and BODY flags are equivalent to specifying the `RETURN=` and `BODY=` parameters in the ESMTP `MAIL` command.

If none of the RETURN flags are specified, the mailer generating the report decides whether just the headers or the full message will be returned. In the case of reports generated by PowerMTA, only the headers are returned.

If none of the BODY flags are specified, the default is 8 bits.

SUBMIT_SetOriginator

Sets the message's originator

```
BOOL SUBMIT_SetOriginator(SUBMIT_CTX* ctx, const char* address)
```

Arguments:

`ctx`
>   Submission context
`address`
>   Originator's e-mail address

Description:

Sets the message's originator. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

SUBMIT_AddRecipient

Adds a recipient

```
BOOL SUBMIT_AddRecipient(SUBMIT_CTX* ctx, const char* address,
    unsigned flags)
```

Arguments:

`ctx`
>   Submission context
`address`
>   Recipient's e-mail address
`flags`
>   A combination of:

319

SUBMIT_RCPT_NOTIFY_SUCCESS	Specifies that a delivery report should be sent in case the delivery succeeds.
SUBMIT_RCPT_NOTIFY_FAILURE	Specifies that a delivery report should be sent in case the delivery fails.
SUBMIT_RCPT_NOTIFY_DELAY	Specifies that a delivery report should be sent in case the delivery is delayed.
SUBMIT_RCPT_NOTIFY_NEVER	Specifies that a no delivery report should be sent.
SUBMIT_RCPT_CHECK_VALIDITY	Specifies that the e-mail address should be checked for validity.

Description:

Adds a recipient to the current message. In case of a failure, zero is returned and the error text can be obtained by calling SUBMIT_GetLastError.

The NOTIFY flags are equivalent to specifying the DSN NOTIFY= parameter in the ESMTP RCPT command. If any of these are specified, they are used by PowerMTA as well as passed on to the receiving mailer — as mandated by DSN.

If none of the NOTIFY flags are specified, the mailer's default is used: if the message has not yet been transferred to another mailer, PowerMTA notifies of failures only. If the message has been passed to a different mailer, that mailer decides what to do. To prevent PowerMTA from sending a bounce (delivery failure notification), you must specify SUBMIT_RCPT_NOTIFY_NEVER with no other NOTIFY flags. Naturally, if no originator address has been specified, no notifications can be sent (or requested).

SUBMIT_AddHeader

Adds a header

```
BOOL SUBMIT_AddHeader(SUBMIT_CTX* ctx, const char* keyword,
    const char* content)
```

Arguments:

ctx

 Submission context

keyword

 Name of header to add

content

 Content of header

Description:

Adds a header line to the current message. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

SUBMIT_AddContentLine

Adds a content line

```
BOOL SUBMIT_AddContentLine(SUBMIT_CTX* ctx, const char* line)
```

Arguments:

`ctx`
>   Submission context

`line`
>   Null-terminated line to add

Description:

Adds a line to the current message's body. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

The line passed should be a standard C string, i.e., a '\0'-terminated array of characters. It *may not* contain any carriage return or line feed characters — PowerMTA appends these automatically.

SUBMIT_AddContentLineLen

Adds a content line of known length

```
BOOL SUBMIT_AddContentLineLen(SUBMIT_CTX* ctx, const char* line,
    unsigned length)
```

Arguments:

`ctx`
>   Submission context

`line`
>   Line to add

`length`
>   Length of line

Description:

Adds a line to the current message's body. In case of a failure, zero is returned and the

321

error text can be obtained by calling `SUBMIT_GetLastError`.

The line passed should be an array of characters, not necessarily '\0'-terminated. It *may not* contain any carriage return or line feed characters — PowerMTA appends these automatically.

SUBMIT_AddContentBlock

Adds a block of content

```
BOOL SUBMIT_AddContentBlock(SUBMIT_CTX* ctx, const char* block,
    unsigned length)
```

Arguments:

`ctx`
> Submission context

`block`
> Data block to add

`length`
> Length of block

Description:

Adds a block of data to the current message's body. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

The data block may contain several lines. Any lines in the block *must* be properly terminated with carriage return and line feed, as mandated by e-mail standards.

SUBMIT_AttachFile

Attaches a file to the message body

```
BOOL SUBMIT_AttachFile(SUBMIT_CTX* ctx, const char* filename,
        const char* contentType)
```

Arguments:

`ctx`
> Submission context

`filename`
> Full path to file to attach

`contentType`
> value `Content-Type` header, e.g. `application/content-stream`

Description:

Attaches a file to the message body. It adds the `Content-Type`, `Content-Disposition` and `Content-Transfer-Encoding` MIME headers, a blank line, and follows with the file contents, base64-encoded.

To attach a file, you should thus add the MIME boundary line just before using `SUBMIT_AttachFile`. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

SUBMIT_FinishMessage

Finishes submitting a message

```
BOOL SUBMIT_FinishMessage(SUBMIT_CTX* ctx)
```

Arguments:

`ctx`

> Submission context

Description:

Finishes the current message and dispatches it. In case of a failure, zero is returned and the error text can be obtained by calling `SUBMIT_GetLastError`.

SUBMIT_GetLastError

Returns the last error occurred

```
char* SUBMIT_GetLastError(SUBMIT_CTX* ctx)
```

Arguments:

`ctx`

> Submission context

Description:

Returns a description of the last error encountered in the specified submission context.

Call this function a) *only* after in fact an error occured, and b) right after the error occured, i.e. without calling any other of the PMTA API functions.

323

A.2. Perl Submission API

The Perl submission API consists of a perl module, (`Submitter.pm`), which provides functions that allow Perl programs to submit messages and write directly to PowerMTA's spool. Since the Perl API is basically a software interface to the C submission API, their inner workings are identical and so we recommend reading through Section B.1.1 before using the Perl API.

A.2.1. Function Reference

new

Creates new submitter object

```
$ctx = new Submitter;
```

Description:
Creates a new submitter object. The submitter is a regular Perl object.

Zero is returned if not enough memory can be allocated.

Reset

Resets the submitter

```
$ctx->Reset ();
```

Description:
Resets the submitter, discarding the message currently being submitted, if not yet finished.

StartMessage

Starts the submission of a new message

```
$ok = $ctx->StartMessage ($envelopeId, $flags);
```

Arguments:

```
envelopeId
```
      Envelope ID for the message.
```
flags
```
      A combination of:

MSG_RETURN_HEADERS	Specifies that any reports (e.g., bounces) should include only the message's headers.
MSG_RETURN_FULL	Specifies that any reports (e.g., bounces) should include the full messsage.
MSG_BODY_7BIT	Specifies that the message being submitted only includes 7-bit data.
MSG_BODY_8BIT	Specifies that the message being submitted contains 8-bit (but not binary) data.
MSG_BODY_BINARY	Specifies that the message being submitted has binary contents. Since binary contents are not yet currently supported, this option is a placeholder only. To provide binary content, you must first encode it (e.g., using base64 encoding).
MSG_OPT_VERP	Enables the VERP (Variable Envelope Return Path) extension. This non-standard SMTP extension is described in http://www.port25.com/support/rfcs/draft-varshavchik-verp-smtpext-03.txt.

Description:

Starts the submission of a new message. In case of a failure, zero is returned and the error text can be obtained by calling GetLastError.

The RETURN and BODY flags are equivalent to specifying the RETURN= and BODY= parameters in the ESMTP MAIL command.

If none of the RETURN flags are specified, the mailer generating the report decides whether just the headers or the full message will be returned. In the case of reports generated by PowerMTA, only the headers are returned.

If none of the BODY flags are specified, the default is 8 bits.

SetOriginator

Sets the message's originator

```
$ok = $ctx->SetOriginator ($address);
```

Arguments:

325

```
address
```
      Originator's e-mail address, such as `'support@port25.com'`

Description:

Sets the message's originator. In case of a failure, zero is returned and the error text can be obtained by calling `GetLastError`.

AddRecipient

Adds a recipient

```
$ok = $ctx->AddRecipient ($address, $flags);
```

Arguments:

```
address
```
      Recipient's e-mail address, such as `'support@port25.com'`
```
flags
```
      A combination of:

`RCPT_NOTIFY_SUCCESS`	Specifies that a delivery report should be sent in case the delivery succeeds.
`RCPT_NOTIFY_FAILURE`	Specifies that a delivery report should be sent in case the delivery fails.
`RCPT_NOTIFY_DELAY`	Specifies that a delivery report should be sent in case the delivery is delayed.
`RCPT_NOTIFY_NEVER`	Specifies that a no delivery report should be sent.
`RCPT_CHECK_VALIDITY`	Specifies that the e-mail address should be checked for validity.

Description:

Adds a recipient to the current message. In case of a failure, zero is returned and the error text can be obtained by calling `GetLastError`.

The NOTIFY flags are equivalent to specifying the DSN `NOTIFY=` parameter in the ESMTP `RCPT` command. If any of these are specified, they are used by PowerMTA as well as passed on to the receiving mailer — as mandated by DSN.

If none of the NOTIFY flags are specified, the mailer's default is used: if the message has not yet been transferred to another mailer, PowerMTA notifies of failures only. If the message has been passed to a different mailer, that mailer decides what to do. To prevent PowerMTA from sending a bounce (delivery failure notification), you must

specify `RCPT_NOTIFY_NEVER` with no other NOTIFY flags. Naturally, if no originator address has been specified, no notifications can be sent (or requested).

AddHeader

Adds a header

```
$ok = $ctx->AddHeader ($keyword, $content);
```

Arguments:

```
keyword
```
  Name of header to add, such as `'To'`
```
content
```
  Content of header, such as `'All subscribers <list-all@my.company.com>'`

Description:

Adds a header line to the current message. In case of a failure, zero is returned and the error text can be obtained by calling `GetLastError`.

AddContentLine

Adds a content line

```
$ok = $ctx->AddContentLine ($line);
```

Arguments:

```
line
```
  Line to add

Description:

Adds a line to the current message's body. In case of a failure, zero is returned and the error text can be obtained by calling `GetLastError`.

The line passed should be a string, which *may not* contain any carriage return or line feed characters — PowerMTA appends these automatically.

AddContentBlock

Adds a block of content

```
$ok = $ctx->AddContentBlock ($block);
```

327

Arguments:

`block`
    Data block to add

Description:
Adds a block of data to the current message's body. In case of a failure, zero is returned and the error text can be obtained by calling `GetLastError`.

The data block may contain several lines. Any lines in the block *must* be properly terminated with carriage return and line feed, as mandated by e-mail standards.

FinishMessage
Finishes message and dispatches it
```
$ok = $ctx->FinishMessage ();
```

Description:
Finishes the current message and dispatches it. In case of a failure, zero is returned and the error text can be obtained by calling `GetLastError`.

GetLastError
Returns a description of the last error
```
$error = $ctx->GetLastError ();
```

Description:
Returns the description of the last error occurred, if any.

Call this function a) *only* after in fact an error occured, and b) right after the error occured, i.e. without calling any other of the PMTA API functions.

Acknowledgments

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. PCRE is available from ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

PowerMTA uses, and may in its distribution kits include, OpenSSL. The OpenSSL license requires that we reproduce it here:

```
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
```

```
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ======================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
```

330

```
 *     "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

PowerMTA uses, and may in its distribution kits include, ICU. The ICU license requires that we reproduce it here:

```
COPYRIGHT AND PERMISSION NOTICE (ICU 58 and later)

Copyright © 1991-2016 Unicode, Inc. All rights reserved.
Distributed under the Terms of Use in
http://www.unicode.org/copyright.html

Permission is hereby granted, free of charge, to any person obtaining
a copy of the Unicode data files and any associated documentation
(the "Data Files") or Unicode software and any associated documentation
(the "Software") to deal in the Data Files or Software
without restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, and/or sell copies of
the Data Files or Software, and to permit persons to whom the Data
Files
or Software are furnished to do so, provided that either
(a) this copyright and permission notice appear with all copies
of the Data Files or Software, or
(b) this copyright and permission notice appear in associated
Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT OF THIRD PARTY RIGHTS.
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS
NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR
CONSEQUENTIAL
DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,
```

DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder
shall not be used in advertising or otherwise to promote the sale,
use or other dealings in these Data Files or Software without prior
written authorization of the copyright holder.

Third-Party Software Licenses

This section contains third-party software notices and/or additional
terms for licensed third-party software components included within ICU
libraries.

1. ICU License - ICU 1.8.1 to ICU 57.1

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2016 International Business Machines Corporation and
others
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, and/or sell copies of the Software, and to permit persons
to whom the Software is furnished to do so, provided that the above
copyright notice(s) and this permission notice appear in all copies of
the Software and that both the above copyright notice(s) and this
permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT
OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY
SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER
RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF
CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder
shall not be used in advertising or otherwise to promote the sale, use
or other dealings in this Software without prior written authorization
of the copyright holder.

All trademarks and registered trademarks mentioned herein are the
property of their respective owners.

2. Chinese/Japanese Word Break Dictionary Data (cjdict.txt)

```
#      The Google Chrome software developed by Google is licensed under
# the BSD license. Other software included in this distribution is
# provided under other licenses, as set forth below.
#
#  The BSD License
#  http://opensource.org/licenses/bsd-license.php
#  Copyright (C) 2006-2008, Google Inc.
#
#  All rights reserved.
#
#  Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
are met:
#
#  Redistributions of source code must retain the above copyright
notice,
# this list of conditions and the following disclaimer.
#  Redistributions in binary form must reproduce the above
# copyright notice, this list of conditions and the following
# disclaimer in the documentation and/or other materials provided with
# the distribution.
#  Neither the name of  Google Inc. nor the names of its
# contributors may be used to endorse or promote products derived from
# this software without specific prior written permission.
#
#
#  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
# CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
# INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
# BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
# LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
#  The word list in cjdict.txt are generated by combining three word
lists
# listed below with further processing for compound word breaking. The
# frequency is generated with an iterative training against Google web
# corpora.
#
#  * Libtabe (Chinese)
#    - https://sourceforge.net/project/?group_id=1519
#    - Its license terms and conditions are shown below.
#
#  * IPADIC (Japanese)
#    - http://chasen.aist-nara.ac.jp/chasen/distribution.html
#    - Its license terms and conditions are shown below.
```

333

```
#
#   ---------COPYING.libtabe ---- BEGIN--------------------
#
#   /*
#    * Copyrighy (c) 1999 TaBE Project.
#    * Copyright (c) 1999 Pai-Hsiang Hsiao.
#    * All rights reserved.
#    *
#    * Redistribution and use in source and binary forms, with or
without
#    * modification, are permitted provided that the following
conditions
#    * are met:
#    *
#    * . Redistributions of source code must retain the above copyright
#    *   notice, this list of conditions and the following disclaimer.
#    * . Redistributions in binary form must reproduce the above
copyright
#    *   notice, this list of conditions and the following disclaimer
in
#    *   the documentation and/or other materials provided with the
#    *   distribution.
#    * . Neither the name of the TaBE Project nor the names of its
#    *   contributors may be used to endorse or promote products
derived
#    *   from this software without specific prior written permission.
#    *
#    * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
#    * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
NOT
#    * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS
#    * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#    * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#    * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR
#    * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
#    * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT,
#    * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#    * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
#    * OF THE POSSIBILITY OF SUCH DAMAGE.
#    */
#
#   /*
#    * Copyright (c) 1999 Computer Systems and Communication Lab,
#    *                    Institute of Information Science, Academia
#    *          *                 Sinica. All rights reserved.
#    *
```

334

```
#    * Redistribution and use in source and binary forms, with or
without
#    * modification, are permitted provided that the following
conditions
#    * are met:
#    *
#    * . Redistributions of source code must retain the above copyright
#    *   notice, this list of conditions and the following disclaimer.
#    * . Redistributions in binary form must reproduce the above
copyright
#    *   notice, this list of conditions and the following disclaimer
in
#    *   the documentation and/or other materials provided with the
#    *   distribution.
#    * . Neither the name of the Computer Systems and Communication Lab
#    *   nor the names of its contributors may be used to endorse or
#    *   promote products derived from this software without specific
#    *   prior written permission.
#    *
#    * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
#    * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
NOT
#    * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS
#    * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#    * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#    * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR
#    * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
#    * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT,
#    * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#    * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
#    * OF THE POSSIBILITY OF SUCH DAMAGE.
#    */
#
#  Copyright 1996 Chih-Hao Tsai @ Beckman Institute,
#      University of Illinois
#  c-tsai4@uiuc.edu  http://casper.beckman.uiuc.edu/~c-tsai4
#
#  ---------------COPYING.libtabe-----END----------------------------
---
 #
 #
 #  --------------COPYING.ipadic-----BEGIN--------------------------
---
 #
 #  Copyright 2000, 2001, 2002, 2003 Nara Institute of Science
 #  and Technology.  All Rights Reserved.
 #
```

```
#   Use, reproduction, and distribution of this software is permitted.
#   Any copy of this software, whether in its original form or
modified,
#   must include both the above copyright notice and the following
#   paragraphs.
#
#   Nara Institute of Science and Technology (NAIST),
#   the copyright holders, disclaims all warranties with regard to this
#   software, including all implied warranties of merchantability and
#   fitness, in no event shall NAIST be liable for
#   any special, indirect or consequential damages or any damages
#   whatsoever resulting from loss of use, data or profits, whether in
an
#   action of contract, negligence or other tortuous action, arising
out
#   of or in connection with the use or performance of this software.
#
#   A large portion of the dictionary entries
#   originate from ICOT Free Software.  The following conditions for
ICOT
#   Free Software applies to the current dictionary as well.
#
#   Each User may also freely distribute the Program, whether in its
#   original form or modified, to any third party or parties, PROVIDED
#   that the provisions of Section 3 ("NO WARRANTY") will ALWAYS appear
#   on, or be attached to, the Program, which is distributed
substantially
#   in the same form as set out herein and that such intended
#   distribution, if actually made, will neither violate or otherwise
#   contravene any of the laws and regulations of the countries having
#   jurisdiction over the User or the intended distribution itself.
#
#   NO WARRANTY
#
#   The program was produced on an experimental basis in the course of
the
#   research and development conducted during the project and is
provided
#   to users as so produced on an experimental basis.  Accordingly, the
#   program is provided without any warranty whatsoever, whether
express,
#   implied, statutory or otherwise.  The term "warranty" used herein
#   includes, but is not limited to, any warranty of the quality,
#   performance, merchantability and fitness for a particular purpose
of
#   the program and the nonexistence of any infringement or violation
of
#   any right of any third party.
#
#   Each user of the program will agree and understand, and be deemed
to
#   have agreed and understood, that there is no warranty whatsoever
for
#   the program and, accordingly, the entire risk arising from or
```

336

```
 #  otherwise connected with the program is assumed by the user.
 #
 #  Therefore, neither ICOT, the copyright holder, or any other
 #  organization that participated in or was otherwise related to the
 #  development of the program and their respective officials,
directors,
 #  officers and other employees shall be held liable for any and all
 #  damages, including, without limitation, general, special,
incidental
 #  and consequential damages, arising out of or otherwise in
connection
 #  with the use or inability to use the program or any product,
material
 #  or result produced or otherwise obtained by using the program,
 #  regardless of whether they have been advised of, or otherwise had
 #  knowledge of, the possibility of such damages at any time during
the
 #  project or thereafter.  Each user will be deemed to have agreed to
the
 #  foregoing by his or her commencement of use of the program.  The
term
 #  "use" as used herein includes, but is not limited to, the use,
 #  modification, copying and distribution of the program and the
 #  production of secondary products from the program.
 #
 #  In the case where the program, whether in its original form or
 #  modified, was distributed or delivered to or received by a user
from
 #  any person, organization or entity other than ICOT, unless it makes
or
 #  grants independently of ICOT any specific warranty to the user in
 #  writing, such person, organization or entity, will also be exempted
 #  from and not be held liable to the user for any such damages as
noted
 #  above as far as the program is concerned.
 #
 #  ---------------COPYING.ipadic-----END----------------------------
----


3. Lao Word Break Dictionary Data (laodict.txt)

 #  Copyright (c) 2013 International Business Machines Corporation
 #  and others. All Rights Reserved.
 #
 # Project: http://code.google.com/p/lao-dictionary/
 # Dictionary: http://lao-dictionary.googlecode.com/git/Lao-
Dictionary.txt
 # License: http://lao-dictionary.googlecode.com/git/Lao-Dictionary-
LICENSE.txt
 #               (copied below)
 #
 #  This file is derived from the above dictionary, with slight
 #  modifications.
```

```
 #  -----------------------------------------------------------------------
---
 #  Copyright (C) 2013 Brian Eugene Wilson, Robert Martin Campbell.
 #  All rights reserved.
 #
 #  Redistribution and use in source and binary forms, with or without
 #  modification,
 #  are permitted provided that the following conditions are met:
 #
 #
 # Redistributions of source code must retain the above copyright
notice, this
 #  list of conditions and the following disclaimer. Redistributions in
 #  binary form must reproduce the above copyright notice, this list of
 #  conditions and the following disclaimer in the documentation and/or
 #  other materials provided with the distribution.
 #
 #
 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 # "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 # FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 # COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 # INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 # (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 # SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 # HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 # STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 # ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 # OF THE POSSIBILITY OF SUCH DAMAGE.
 #  -----------------------------------------------------------------------
-------

4. Burmese Word Break Dictionary Data (burmesedict.txt)

 #  Copyright (c) 2014 International Business Machines Corporation
 #  and others. All Rights Reserved.
 #
 #  This list is part of a project hosted at:
 #    github.com/kanyawtech/myanmar-karen-word-lists
 #
 #  -----------------------------------------------------------------------
-------
 #  Copyright (c) 2013, LeRoy Benjamin Sharon
 #  All rights reserved.
 #
 #  Redistribution and use in source and binary forms, with or without
 #  modification, are permitted provided that the following conditions
 #  are met: Redistributions of source code must retain the above
 #  copyright notice, this list of conditions and the following
 #  disclaimer.  Redistributions in binary form must reproduce the
 #  above copyright notice, this list of conditions and the following
 #  disclaimer in the documentation and/or other materials provided
 #  with the distribution.
```

```
#
#    Neither the name Myanmar Karen Word Lists, nor the names of its
#    contributors may be used to endorse or promote products derived
#    from this software without specific prior written permission.
#
#  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
#  CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
#  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
#  MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
#  BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
#  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
#  TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
#  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
#  ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
#  TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
#  THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
#  SUCH DAMAGE.
#  ------------------------------------------------------------------
-------
```

5. Time Zone Database

   ICU uses the public domain data and code derived from Time Zone
Database for its time zone support. The ownership of the TZ database
is explained in BCP 175: Procedure for Maintaining the Time Zone
Database section 7.

```
 # 7.  Database Ownership
 #
 #    The TZ database itself is not an IETF Contribution or an IETF
 #    document.  Rather it is a pre-existing and regularly updated work
 #    that is in the public domain, and is intended to remain in the
 #    public domain.  Therefore, BCPs 78 [RFC5378] and 79 [RFC3979] do
 #    not apply to the TZ Database or contributions that individuals
make
 #    to it.  Should any claims be made and substantiated against the
TZ
 #    Database, the organization that is providing the IANA
 #    Considerations defined in this RFC, under the memorandum of
 #    understanding with the IETF, currently ICANN, may act in
accordance
 #    with all competent court orders.  No ownership claims will be
made
 #    by ICANN or the IETF Trust on the database or the code.  Any
person
 #    making a contribution to the database or code waives all rights
to
 #    future claims in that contribution or in the TZ Database.
```

339

DIRECTIVES INDEX

acct-file

address-list

aliases

alphabetical

bounce-processor

deprecated

domain

345

feedback-loop-processor

global

pattern-list

smtp-user

source

source {pickup}