SPEC OSG Mailserver Subcommittee
SPECmail2008 Benchmark Architecture White Paper

Revision: v1.1
Date: 2 June 2008

1. Introduction

1.1 Overview

SPECmail2008 is a software benchmark designed to measure a system's ability to act as an enterprise mail server servicing email requests, based on the Internet standard protocols SMTP and IMAP4. The benchmark concentrates on the workload encountered by corporate mail servers, with an overall user count in the range of 150 to 10,000 (or more) users. It models IMAP business users accessing IMAP servers over fast local area networks (LAN) instead of broadband, WAN or dialup access speeds.

SPECmail2008 has been developed by the Standard Performance Evaluation Corporation (SPEC), a non-profit group of computer vendors, system integrators, universities, research organizations, publishers, and consultants.

This paper discusses the benchmark principles and architecture, and the rationale behind the key design decisions. It also outlines the workload used in the benchmark, and the general steps needed to run a benchmark. However those aspects are covered in more detail in other documents.

1.2 Organization of this Paper

Chapter 2 discusses the basic goals and non-goals of the benchmark.

Chapter 3 introduces the performance metric of SPECmail2008 – IMAP sessions per hour - and how it relates to the transaction mix imposed on the system under test..

Chapter 4 explains the benchmark workload - how it was derived, how it translates into configuration parameters for the benchmark tool and size calculations for planning a benchmark, and how it relates to the benchmark metric.

Chapter 5 discusses some detail of aspects of the workload generation, namely the exact workload put on the server, and how the benchmark simulates communication with remote mail servers.

Chapter 6 defines the quality of service requirements of this benchmark.

Chapter 7 lists the references and sources (not cited elsewhere).

1.3 Related Documents

- SPECmail2008 Run and Reporting Rules
- Workload Analysis for Enterprise Mail Servers
- SPECmail2008 User Guide
- SPECmail2008 Sample Result Disclosure
- SPECmail2008 FAQ

All documents can be obtained from the mail server working group's home page http://pro.spec.org/private/osg/mail_server.

1.4 Run and Reporting Rules

The Run and Reporting Rules for the SPECmail2008 benchmark are spelled out in a separate document. They ensure execution of the benchmark in a controlled environment. The goal is repeatability by third parties with reasonable resources and knowledge. The rules maximize the comparability of results, leveling the playing field as much as possible. They also define which information needs to be included in published results, and which supporting information needs to be submitted to the SPEC community for potential review.

Under the terms of the SPEC license, SPECmail2008 results may not be publicly reported unless they are run in compliance with the Run and Reporting Rules. Results published at the SPEC web site have been reviewed and approved by the SPEC Mail Server committee. For more information on publishing results at the SPEC web site, please send e-mail to: info@spec.org. The Run and Reporting Rules may be found on the SPEC web site; they are also part of the SPECmail2008 distribution kit.

2. Design of the SPECmail2008 Benchmark

SPECmail2008 benchmark tests the capacity of a system as an e-mail service that processes requests according to the Internet standard mail protocols ***SMTP*** (RFC 821) and ***IMAP4*** (RFC 2040). The SMTP protocol is the standard for sending e-mail from clients (users) to servers and between e-mail servers. The IMAP4 protocol allows users to access and retrieve messages from their message store. The mail server can consist of one host or a group of hosts that act as a single, logical entity – usually represented by a single e-mail domain.

The SPECmail_MSEnt2008 metric’s user model describes a corporate employee that uses one of the popular IMAP4 clients to access either a mail server located within a local area network (LAN), or an outsourced e-mail service across a high-speed network connection (MAN). The details of an enterprise-type user's behavior will be discussed in a later section in this paper that covers the SMTP and IMAP work load profiles.

In this benchmark, both the Mail Server User behavior and the IMAP4 e-mail client software vary greatly. Therefore, it is very important to identify and distinguish actual IMAP4 client's human initiated actions from automated actions performed on behalf of each user. The specific combination of these behavior types determines how many users a mail server can handle.

SPECmail2008 simulates the work loads of four types of IMAP4 e-mail clients, defined in a fixed proportion across the user population. The benchmark observes the mail server behavior under that load. It enforces the adherence to a required level of quality of service. The goal is to simulate realistic mail server operation, and maximize the usefulness of the benchmark results as guidelines for actual sizing decisions.

2.1 <u>Requirements</u> <u>and</u> <u>Goals</u>

The key goal of SPECmail2008 is to show mail server performance in a realistic context. This means

Appropriate Mail Store	A mail server which handles the transaction load of a corporate user base needs to hold the mail data for the same amount of users. This includes a mailbox per user, as well as a defined number of folders and messages for each user.
Mail Store Folder Structures	The benchmark includes the construction and pre-population of multiple folders (IMAP *mailboxes*) and subfolders as part of the compliance requirements. Since IMAP is a server-centric storage service, access and manipulation of folders plays a critical part of the overall workload.
Message MIME Structures	The benchmark includes the construction and manipulation of e-mail messages with both simple and more complex MIME (messages with attachments) structures. Unlike POP3 servers, IMAP clients understand and can manipulate multi-part MIME messages.
Arrival Rate	Message activity requests are being issued against the server according to behavior patterns and arrival rate modeled after real world observations.
Access over Local Area and/or Broadband Networks	All users will access the IMAP and SMTP servers via internal corporate or broadband network speeds.
Workload According to Real World Data	The simulated workload (such as messages sent and received per day, the message size distribution, the folder structure distribution, the mailbox access frequency, message manipulation) is based on measurements derived from multiple data sources.
Peak Hour Simulation	Mail traffic is distributed unevenly over the day. Sizing must focus on coping with the peak load. Therefore, the benchmark mail server load simulates the peak hour of the day.
Realistic Operation	The mail server is required to operate realistically, e.g. perform at least a defined level of logging which many ISPs would use in practice. As is the rule in all SPEC benchmarks, no benchmark-specific optimizations are allowed.

2.2 Excluded Goals

Explicitly excluded goals and limitations in the scope of SPECmail2008 are:

Restricted client IP Range	The benchmark does not require that there be 100's of small, remotely connected email clients. Instead, it allows simulation of the email clients on a small number of server-sized, locally connected client systems, in order to make the benchmark execution practical.
Administrative Overhead	The benchmark does not include administrational activities like on-line backup and account provisioning. These are hard to standardize, and - more importantly - they do not necessarily happen during the peak hour of daily operation.
Provisioning Overhead	Besides provisioning, the benchmark does not include modification to account data or deletion of accounts. These activities, although commonplace in the day-to-day operation of a mail server, are seen as testing the directory service rather than the mail server itself. SPECmail2008 emphasizes mail server benchmarking, and includes a directory server in the system-under-test only to the minimum extent necessary to handle normal mail flow.
Pure Relay SMTP Traffic	The benchmark does not cover relay operation (forwarding of incoming SMTP traffic to other, remote MTAs). It does include simulation of messages from local users to remote MTAs, as well as incoming mail from remote MTAs. Relay operation is generally not allowed on enterprise e-mail systems.
High Availability Overhead	There is no requirement for high availability or disaster recovery measures; this was beyond the scope of the benchmark.
Extra Content Filter/Processing, Security	The mail server is not required to perform any extended features, like virus scanning, spam filtering and other security measures. This was beyond the scope of the benchmark.
SMTP Streamlining	SMTP sessions can normally send one or several messages per session. The benchmark restricts itself to one message. The collected SMTP log data shows that this behavior dominates internally and externally generated e-mail traffic.

3. Benchmark Metric: **SPECmail_MSEnt2008**

SPECmail2008 Enterprise IMAP4 Sessions Per Hour

The basis of the benchmark's performance metric is *capacity at acceptable quality of service (QoS)*. The benchmark determines *Acceptable QoS* by measuring the interactive response time of the mail server to each protocol step (see Chapter 6). The boundary where a one or more critical states exceeds the *Acceptable QoS* determines how many users the SUT supports.

The IMAP protocol allows many combinations of session behavior and duration – very unlike a typical POP3 session used in SPECmail2001. Next, IMAP clients use more than one connection into an IMAP server to perform different tasks. Lastly, each IMAP user generate many IMAP sessions – both in parallel and serially over the peak hour period. The number of extra sessions varies by IMAP client software. Therefore the number of concurrent IMAP4 sessions is often some multiple of the actual active IMAP users. The actual count is determined by both active users and the specific distribution of IMAP client types.

The SPECmail2008 benchmark uses a specific distribution of the four (4) possible IMAP4 client types that a mail server must support during the peak hour. This transaction mix is defined later on in this document. In general, the following conditions exist:

User Mail Store	Each user has at least three (3) folders – Inbox, Trash, Sent_Mail Each user has existing messages stored in these folders
SMTP Traffic	Each user send 5 messages during Peak Hour to 3.0.6 average recipients Each user receives 5.06 messages during Peak Hour
IMAP Session	Each user establishes at least one long lived session before the peak hour Some short lived sessions are initiated by human actions Some short lived sessions are initiated by automated tasks
Folder Activity	Each user polls one or more folders in each session at least once Some users will created new and/or delete old folders Some users will move messages between folders
Message Activity	Each user will retrieve all new messages during the Peak Hour Some users will also retrieve older messages Each user might delete some messages, both old and new A few users might search a folder

This metric cannot be compared to any other similarly named benchmark metric which does not follow exactly that workload definition and the same execution rules. Every aspect of each may affect the benchmark outcome.

4. IMAP4 Benchmark Decisions

4.1 Command Set

The various versions of IMAP4 RFCs as well as any number of optional extended commands made an analysis of the IMAP4 variants problematic. Because of the variety of client and server combinations, the subcommittee used the following guidelines to determine the IMAP commands used.

Restrict Command Set to RFC 2040	The decision was to stay with a very strict interpretation of RFC 2040 and ignore later command set extensions. All e-mail servers and clients supported the base command set defined by RFC 2040. However, support for extended IMAP commands varied by both mail server and mail clients.
Reduce Parameters to Most Common Forms	Identify the most common forms of various IMAP commands. Identify the most frequent parameter combinations, ignoring sequence, to these commands. Map certain parameters to an equivalent form.
Ignored Commands	Ignore certain extended IMAP4 commands that affect/query meta-data information, and is not directly related to messages or folders. These include `CAPABILITY`, `GETACL`,

	`MYRIGHT`, and `NAMESPACE`.
Map Extended Commands to Base RFC Version	Certain extended IMAP4 commands duplicate functionality found in the original RFC 2040 command set. These commands were mapped accordingly: `AUTHENTICATE` to `LOGIN` and `IDLE/DONE` combination to `NOOP`.

4.2 Mail Store Structures

The various versions of IMAP4 RFCs as well as any number of optional extended commands made this analysis mandatory. The decision was to stay with a very strict interpretation of RFC 2040 and treat some of the extended commands in the following manner.

Message Set References	Some commands used message identifiers that were a combination of a range (A:C) and a discrete set (A,C,F). These will be mapped to just a message identifier range (A:C).
Specified Headers vs. Number of Headers	The message retrieval requests for specific headers incur the same cost regardless of which headers are retrieved. Therefore, certain headers that are not listed in RFC821 will be generalized into a small subset.
Restrict Folder Depth References to Five (5) Levels	The extremely low probability rates beyond Level 5 were collapsed into a single row at Level 5.
Reserved Folder Names	All IMAP4 client types referenced three folder names: Inbox, Inbox.Trash, Inbox.Sent. These will be created for all users as part of the mail store initialization step.
Message MIME Structure and Part Sizes Predominate	Unlike POP3 where messages are retrieved in whole, a significant number of IMAP4 commands retrieved specific MIME attachments. This means that message structures must exist. Once this condition exists, it is almost impossible to comply with the message size distributions defined by the SMTP logs. The choice was made to let message structure and attachment sizes be the primary factors, overriding raw message sizes.
Restrict MIME Depths to Five (5) Levels	As with the folder nesting levels, depths below five (5) were extremely low probability. These were mapped to Level 5.

4.3 Compliant Run

The SPECmail2008 benchmark compliant conditions are as follows:

User Count	Set to at least **200** users in order to meet the folder structure distribution.
Quality of Service	remains at **5** seconds for "simple" commands

5. Workload

The SPECmail2008 workload has two parts: the pre-population of the mail store with folders, sub-folders and messages, as well as the transaction workload during runtime. Both depend on the targeted benchmark rating.

It may be helpful at this point to list the basic steps of running a SPECmail2008 benchmark.

Benchmark Step	**What Happens**
Initialize Mail Store	Pre-populate the mail store according to the defined folder and message MIME distributions. An alternative is to restore a backup copy of a previously initialized mail server.
Restart SUT	Start from a cold system (processes, file systems, databases), so that all operating system caches are clean.
Verify Mail Store Compliance	Verify mail store has the desired folder, message quantity and message MIME structure distributions across all mailboxes.
Gather Mailbox Context	Benchmark gathers detailed information about actual mail store folder names and valid messages to be used later to generate valid IMAP commands.
Ramp-up Period	Benchmark runs for a configurable time at peak hour load levels, but no measurements are recorded. During this time, the benchmark establishes the pre-existing Command Sequence 1 and 2 sessions.
Peak Hour Load Test at 100%t	Internal results counters are cleared (except existing IMAP sessions) and the benchmark manager starts recording results returned by the load generators for the work load period.
Ramp-down Period	Benchmark finishes all existing IMAP sessions but does not initiate new sessions. Primary sessions (long lived) sessions log out of the mail server.
Record Results	Tabulate all collected results and record certain statistics to the official results file.
Report Generation	Use distributed reporter to generate human readable form of the benchmark test results.

The actual process is a bit more complex - refer to the Run and Reporting Rules for details.

5.1 Basis of Workload Definition

The workload profile has been determined based on actual SMTP and IMAP4 server log files gathered from multiple corporate sources. The IMAP4 data covers e-email traffic from two (2) universities, two (2) corporate e-mail servers and one (1) outsourced e-mail service.

The SMTP workload is extracted MTA log files and processed for arrival rates, recipient counts, message sizes and routing (local vs. remote). The recipient distribution includes exploded mailing lists as well as individually addressed recipients. (Sources: Mirapoint, Openwave)

The mailbox structures and contents were derived from the list of actual folders found during a mail server mail store survey. Every user's mail store folders were listed along with a count of the messages and subfolders inside. (Sources: Mirapoint, Openwave, Sun Microsystems)

The message MIME structures and content types were derived from a complete snapshot of an E-mail server's message structures. (Source: Sun Microsystems)

5.2 Non-Transaction Related Definitions

The typical Enterprise E-Mail server with IMAP users holds user messages stored in one or more folders. Unlike the POP3 e-mail clients, IMAP e-mail clients work with both new and existing messages. This means the folder structures and messages must already exist before the work load can start.

Mail Store Structures	The SPECmail2008 benchmark defines a mail store structure model that replicates a complex mail store structure, derived from data collected from Sun and Openwave. Some folders that will have up to two thousand (2000) messages inside. The pre-population of the mail server ensures that a system can handle the transaction and storage load for 200 Enterprise users. The pre-population consists of a folder distribution and a message distribution across all mailboxes. The runtime load defines the number and kind of transactions that will be issued to the server during the benchmark run.
Mail Server Storage Growth	The SPECmail2008 benchmark is designed to generate a steady mail server state: over time. However unlike the SPECmail2001 benchmark, it is not storage neutral. Based on the collected IMAP session samples, fewer messages were deleted than created during the workload period. Note: message insertion and deletion happens in folders that are randomly and independently selected. Therefore, mailbox structures and content change during the load test period.
Message MIME Structures	The IMAP protocol understands the concept of MIME structures and many data samples show references to message substructures. This means these structures must exist to during the peak hour work load.

5.2.1 Folder Structures

A mail server that supports IMAP is likely to support a hierarchy of several mailboxes (also known folders) in addition to the default INBOX mailbox for each user. Below are several distributions to construct the structure of mailboxes contained within a mailstore supported by IMAP. The data used is extracted from the three enterprise data samples (Mirapoint, Openwave, Sun).

Configuration Key	**Definition and Value**

`LEVELXFOLDERS[]`	Defines the probability that a mail store has one or more folders, at various depth.
	`# folder with sub-folder distributions from the Mirapoint sample.`
	`LEVELXFOLDERS[0] = "1,34.8%; 2,21.7%; 3,11.6%; 4,7%; 5,2%; 6,2.4%; 7,1.5%; 8,.7%; 9,.7%; 10,.7%; 20,8.1%; 30,3.7%; 40,1.8%; 50,2%; 103,1.3%"`
	`LEVELXFOLDERS[1] = "1,31.4%; 2,12.4%;`

```
3,7.4%; 4,5.6%; 5,4%; 6,2.4%; 7,5%;
9,5.8%; 10,2.6%; 15,7.4%; 20,3.2%;
30,7.2%; 70,3.4%; 200,1.8%; 246,.4%"
LEVELXFOLDERS[2] = "1,43%; 2,14.9%;
3,9.1%; 4,6.8%; 5,3.5%; 6,4.1%; 7,2%;
8,2%; 9,3.3%; 10,1%; 11,1.5%; 12,1.3%;
13,.8%; 14,.3%; 15,.5%; 17,.3%; 19,1.3%;
25,.3%; 26,.8%; 28,1.3%; 29,.3%; 30,.5%;
38,.3%; 39,.2%; 41,.2%; 47,.2%; 61,.2%"
LEVELXFOLDERS[3] = "1,39.6%; 2,12.6%;
3,8.1%; 4,10.8%; 5,2.7%; 6,7.2%; 7,2.7%;
8,.9%; 9,1.8%; 11,1.8%; 12,1%; 13,.9%;
15,.9%; 16,.9%; 19,1.8%; 20,.9%;
22,1.8%; 25,.9%; 26,1.8%; 42,.9%"
LEVELXFOLDERS[4] = "1,36.8%; 2,7.9%;
3,39.5%; 4,5.3%; 6,2.6%; 7,2.6%; 8,5.3%"
LEVELXFOLDERS[5] = "1,100.0%"
LEVELXFOLDERS[6] = "1,100.0%"
LEVELXFOLDERS[7] = "1,100.0%"
```

`LEVELXWITHSUB` `_PART_LEVEL` Defines the probability distribution that a folder at each depth (indicated by the array index) has zero or more subfolders:

```
LEVELXWITHSUB[0] = "0,59%; 1,21.9%;
2,7.5%; 3,3.3%; 4,2%; 5,.4%; 6,1.3%;
7,2%; 8,.4%; 9,.7%; 10,.7%; 11,.4%;
21,.25; 26,.15%"
LEVELXWITHSUB[1] = "0,64%; 1,20.6%;
2,9%; 3,1.4%; 4,1.2%; 5,.8%; 6,1%;
7,.6%; 8,.2%; 9,.2%; 10,.4%; 15,.4%;
19,.2%"
LEVELXWITHSUB[2] = "0,80%; 1,15.7%;
2,2.3%; 3,.8%; 4,1%; 6,.2%"
LEVELXWITHSUB[3] = "0,78.4%; 1,14.4%;
2,3.6%; 3,1.8%; 4,1.8%"
LEVELXWITHSUB[4] = "0,97.4%; 1,2.6%"
LEVELXWITHSUB[5] = "1,100.0%"
LEVELXWITHSUB[6] = "1,100.0%"
LEVELXWITHSUB[7] = "0,100.0%"
```

The folder probability distributions exists for eight levels, but only the first five (5) levels, are used for the message store compliance verification test. The lower levels were excluded from the benchmark because of the extremely small probability that such a level would exist.

5.2.2 Message Construction

The SPECmail2008 benchmark generates test messages on the fly instead of using a set of fixed messages. Construction of each message follows these steps:

Message Construction Step	Explanation
Determine Number of MIME Parts at Top Level	Using the `MIME_TOP_PART_COUNT` distribution, compute how many overall parts make up the message.
Determine Each Part's	Using the `MIME_PART_LEVEL` distribution, compute the

Substructure	number of parts each Top level MIME part contains. No effort is made to distinguish between primary and alternate parts since this work load appears as nearly identical commands to the mail server.
Determine Each Part's Size	Using the `MIME_PART_SIZE` distribution, compute the size of each individual MIME part. This is the primary message size factor.

As described earlier, SPECmail2008 chose to follow message structural and attachment size distributions rather than the total message size distribution used by the earlier SPECmail2001 benchmark. In SPECmail2001, both the e-mail server and POP3 client tend to ignore the actual message MIME structure; recognizing just headers and body components.

IMAP4 e-mail clients understand the concepts of attachments and expect the e-mail server to understand the various message parts. This meant that the e-mail server must evaluate the actual structure of each message. Therefore, message structure and individual attachment sizes affect the actual message size, since the MIME structural description is embedded in the message but not visible to most users.

5.2.2.1 Message Size (Deprecated: MSG_SIZE_DISTRIBUTION)

The SPECmail2001 method created a single level message that met a fixed message size distribution. This was not possible with SPECmail2008 and the subcommittee decided to generate messages according to the various MIME distributions instead of the MSG_SIZE_DISTRIBUTION tables.

5.2.3 Multipurpose Internet Mail Extension (MIME) Profile

MIME is an internet attachment scheme, defined as a formal standard by RFCs 1521, 1522, and 1523. The Sun data set provided detained information about mailbox and message structure. Thus it is the basis for the following probability distribution tables used in the benchmark.

5.2.3.1 MIME Message Construction

The initial processing of all message sizes distinguished between single part sizes and multipart sizes. The SPECmail2008 benchmark prioritizes individual MIME part size over the global message size distribution.

MIME Part Count	Construction Rules
Single (1) Part	• Use "Content-type: text/plain" in message headers • Use subpart content size distribution
Multiple Parts	• Use "Content Type: multipart/mixed; boundary="xxxxxxxxx-counter" in message headers • Use distributions for message part width and depth to help establish the set of multipart message bodies. • Categorize MIME messages to fall into one of these pre-defined multipart buckets. • Use subpart content size distribution to define the sub-part sizes in the fixed pool of pre-defined multipart messages.

The Top-Level Part Count distribution determines the probability that a message has N parts, where N is at least one (1). After computing the number of parts at any one level, the benchmark computes the probability that each message part contains further sub-levels, as defined by the MIME Part Depths distribution. Each message part size and MIME types using the (configuration key) MIME_PART_SIZE and internal MIME Content Type Distributions to build the actual message content.

This table shows the distribution of primary MIME Content Type (not including subtype) of all the parts in the entire sample. It is embedded inside the benchmark code and not configurable.

Content type	Probability	Content type	Probability
TEXT	92.193%	IMAGE	0.888%
APPLICATION	4.265%	AUDIO	0.016%
MESSAGE	2.633%	VIDEO	0.004%

Table 1: MIME Content Type Distribution (Source: Sun)

5.2.3.2 Benchmark MIME Message Distributions

The benchmark uses the MIME Parts, MIME Part Sizes and MIME Depth distribution tables to construct each message stored in the mail store. These configuration keys are fixed and cannot be changed for a compliant run.

Configuration Key	**Definition and Value**
`MIME_TOP_PART_COUNT`	Defines the probability that a message will have one (1) through five (5) MIME parts. Each is either a message text, zero or more attachments, or alternative views of the same message texts (example plain versus rich text versions of the same message): `MIME_TOP_PART_COUNT = "1,75.77%; 2,21.91%; 3,1.99%; 4,0.24%; 5,0.09%"`
`MIME_PART_LEVEL`	Defines the probability distribution of a message having up to five (5) nesting levels in the overall MIME structure: `MIME_PART_LEVEL = "1,91.24%; 2,7.73%; 3,0.87%; 4,0.13%; 5,0.03%"`
`MIME_PART_SIZE`	Defines the probability distribution of individual MIME part sizes: `MIME_PART_SIZE = "2,0.6%; 4,0.1%; 8,0.4%; 16,0.8%; 32,1.8%; 64,4.1%; 128,7.2%; 256,10.5%; 512,15.6%; 1024,13.6%; 2048,13.9%; 4096,13.4%; 8192,8.5%; 16384,4.3%; 32768,2.3%; 65536,1.2%; 131072,0.7%; 262144,0.4%; 524288,0.3%; 1048576,0.2%; 2097152,0.1%"`

5.3 Transaction Workload

The Transaction Workload defines the number and type of transactions issued against the system under test during the measurement phase of the benchmark. It scales linearly with the number of users and with the **SPECmail_MSEnt2008** metric. Its parameters are:

User Mail Store	The overall folder (mailbox) quantity and depths distributions (see prior section)
SMTP	The number of new SMTP sessions established per second and the percent of SMTP traffic between local and remote domains
IMAP	Number of existing IMAP4 sessions present as well as the number of new IMAP4 sessions established per second
Message	The distribution and quantity of existing messages entering the peak hour, using the specified message MIME structure and part size distributions.

5.3.1 SPECmail2008 Enterprise Workload Profile

The definition of the transaction workload starts with an assessment of the per-user, per-day load profile. The following tables show assumptions for that profile, as well as the semantics for the elements in that profile. Names in ALL_UPPER_CASE are actual configuration parameters of the benchmark tool, names using UpperAndLower case are inserted only for editorial purposes.

Configuration Key	**Definition and Value**
`USERNAME_PREFIX`	First part of all test user names used as SMTP addresses and IMAP4 login names. Default: `USERNAME_PREFIX = spec`
`USER_END`	Maximum value appended to `USERNAME_PREFIX` and must be at least 200 larger than `USER_START`: Default: `USER_END = 200`
`USER_START`	Minimum value appended to `USERNAME_PREFIX`: Default: `USER_END = 1`
`USERNAME_AS_PASSWORD`	Use the generated account name as the login password. Default: `USERNAME_AS_PASSWORD = 1 or 0`
`USER_PASSWORD`	Use this value as the login password for all accounts. Example: `USER_PASSWORD = <string>`
`IMAP_SERVER`	Hostname (not IP) of SUT’s IMAP4 server
`IMAP_PORT`	Socket number of SUT’s IMAP4 server
`SMTP_SERVER`	Hostname (not IP) of SUT’s SMTP server
`SMTP_PORT`	Socket number of SUT’s SMTP server
`LOCAL_DOMAIN`	Domain name associated with local accounts
`REMOTE_DOMAIN`	Domain name used to relay message
`CLIENTS`	List of hostname:port pairs representing one or more load generators
`THREADS_PER_CLIENT`	Maximum number of Java threads used by each load generator during initialization, verification and context gather phases. Default: `THREADS_PER_CLIENT = 1`

`LOAD_FACTORS`	Percent of computed load to use as part of the load test period. Default: `LOAD_FACTORS = 100`
`RUN_SECONDS`	How long to run and gather statistics at peak hour loads. Default: `RUN_SECONDS = 3600`
`WARMUP_SECONDS`	How long to run at peak hour loads before gathering statistics. Default: `RUN_SECONDS = 600`

Table 2: Benchmark and SUT Configuration Keys

5.3.2 SMTP Workload Profile

The following table summarizes the compliant SMTP workload used during the load test period.

Configuration Key	Definition and Value
`PEAK_PCT_USERS`	Percent of provisioned users who receive messages during the peak hour (also known as Active users). Default: `PEAK_PCT_USERS = 78%`
`MSG_RECEIVED_PER_PEAK_HOUR`	Number of messages received by Active users during peak hour: Default: `MSG_RECEIVED_PER_PEAK_HOUR = 5.06`
`MSG_RECP_DISTRIBUTION`	Distribution defining number of recipients per message. The average is 3.0.6 Default: `MSG_RECP_DISTRIBUTION = 1,46.3875%; 2,11.00%; 3,9.00%; 4,8.00%; 5,7.00%; 6,6.00%; 7,5.00%; 8,4.00%; 10,2.00%; 11,1.00%; 12,0.30%; 13,0.10%; 14,0.05%; 15,0.05%; 16,0.05%; 30,0.05%; 50,0.01%; 100,0.0025%`
`LOCAL_TO_LOCAL_PCT`	Percent of total messages sent from Local users to Local users Default: `LOCAL_TO_LOCAL_PCT = 56`
`REMOTE_TO_LOCAL_PCT`	Percent of total messages sent from Remote users to Local users Default: `REMOTE_TO_LOCAL_PCT = 31`
`LOCAL_TO_REMOTE_PCT`	Percent of total messages sent from Local users to Remote users Default: `LOCAL_TO_REMOTE_PCT = 13`

Table 3: Peak Hour SMTP Normalized User Profile

5.3.3 SMTP Message Rates

The message inter-arrival time computation uses a simplified model because the total number of new messages tends to be insufficient to fulfill a complex distribution. Therefore, the time between message delivery attempts is computed as the total number of messages to be delivered over the duration of the load test run time, divided by that run time.

SMTP Inter-arrival Time = (*Number of Active Users*) X (*Messages per User*) X (*Recipients per Message*) / (*Load Test Time (s)*)

5.3.4 IMAP Workload Profile

IMAP sessions are fairly complex in nature due to their long life times, the ability to manage multiple folder layers, and the ability to append, retrieve, and delete messages. The SPECmail2008 IMAP Workload is defined by the combination of two categories: client-type and command sequences.

Command Sequence	A series of IMAP commands issued after the initial `LOGIN` command, and terminated either a `LOGOUT` command or an idle session timeout value.
Client Type	A collection of one or more command sequences that characterizes the type of IMAP sessions different IMAP4 clients implement.

The following table describes the criteria for each command-sequence.

Command Sequence	General Characteristic	Comments
1	• Create connection • Perform several operations using a variety of commands (probe folder for new messages, deleting, and moving messages, updating flags, list available folders, appending messages, searching for messages, checkpointing, etc.) • Occasionally probe folders for new messages • Fetch headers if any messages arrived • Occasionally fetch body (whole or parts of body) • Focuses on a specific folder • Does not log out session	**Example Clients**: Netscape (Mozilla), Pine, Mulberry) This is one of the "primary" sessions that tend to stay logged into the IMAP server for many hours or days. Netscape uses UID commands, Pine and Mulberry do not. Probing folders is accomplished by: 1. Netscape: NOOP; UID FETCH n:* (FLAGS) 2. Mulberry: SEARCH UNSEEN; SEARCH DELETED; FETCH 1:m (FLAG ENVELOPE BODYSTRUCTURE, …) 3. Pine: NOOP
2	• Create connection • Perform several operations using a variety of commands (probe folder for new messages, deleting, and moving messages, updating flags, list available folders, appending messages, searching for messages, checkpointing, etc.) • Occasionally fetch headers • Occasionally fetch header and whole body • Does ***not*** focus on a specific folder • Does not log out of session	**Example Clients**: Outlook, Outlook Express, Mulberry This is one of the "primary" sessions that tend to stay logged into the IMAP server for many hours or days. Probing folders is accomplished by these IMAP commands: • UID FETCH n:* (UID, BODY.PEEK[HEADER], …) • UID FETCH 1:n-1 (UID FLAGS)

Command Sequence	General Characteristic	Comments
3	• Create connection • Fetch headers • Fetch whole body • Logout	**Example Clients**: Fetchmail, Outlook Express These sessions are very sporadic and show dependency on results returned from Command Sequence 4.
4	• Create connection • Occasionally probe folders for new messages • Occasionally issue other IMAP commands that does not alter the state of the mailstore (such as UNSUBSCRIBE or LIST) • Sometimes logs out, not always	**Example Clients**: Outlook, Outlook Express, Netscape - periodic or triggered actions These sessions show very automated behavior and are generated at fixed intervals for each user. Probing folders is accomplished by: • Outlook 2002 – Inbox: UID FETCH m:* (UID, BODY.PEEK[HEADER], …); or UID FETCH 1:n (UID FLAGS) • Outlook 2002 – Others: LSUB "" "*"; or STATUS "mailbox name **1**" (UNSEEN); ..; STATUS "mailbox name **n**" (UNSEEN); • 2. Outlook Express: STATUS "mailbox name" (MESSAGES UNSEEN)
5	• Create connection • Occasionally list or probe folders • Perform specific tasks, such as deleting, messages, or appending messages, etc. • Alters the state of the mail store • Logout	**Example Clients**: Mulberry, Netscape These sessions tend to focus on a specific set of tasks and then log out of the IMAP server.

Table 4: IMAP Command Sequence Definition

IMAP4 clients tend to use one or more of the five (5) command sequences, connecting one or more times to the IMAP server. The IMAP4 benchmark emulates four (4) client types, with each client type session cluster representing a single user.

Client Type	Constituent	Comments
1	CS1 CS4	These two (2) command sequences operate independently and concurrently. Some of these clients use a message index number while others use the message UID.
2	CS1 CS4 CS5	These three (3) command sequences operate independently and concurrently. Some of these clients use message index number while others use the message UID.
3	CS2 CS3 CS4	These three (3) command sequences depend on both user and automated actions. The CS2 primary session tend to govern the tasks done in CS4. The CS3 sessions are automated and influences the other two.

Client Type	Constituent	Comments
4	CS2 CS4 CS5	These three (3) command sequences operate independently and in parallel. The client uses message index number instead of message UID.

Table 5: IMAP Client Type Definitions

The compliant run uses the following combination to determine sequencing and dependencies.

Configuration Key	Definition and Value
`PEAK_LOAD_PERCENT`	Percent of the daily IMAP load occurring during the peak hour: Default: `PEAK_LOAD_PERCENT = 32`
`CLIENT_TYPE_DISTRIBUTION`	Defines the probability that a message will have one (1) through five (5) MIME parts. Each is either a message text, zero or more attachments, or alternative views of the same message texts (example plain versus rich text versions of the same message): Default: `CLIENT_TYPE_DISTRIBUTION = "1,31.373%; 3,32.353%; 4,3.922%; 5,2.941%; 13,3.922%; 14,10.784%; 15,1.961%; 24,0.980%; 34,2.941%; 45,2.941%; 134,0.980%; 145,3.922%; 1245,0.980%"`
`CS3_MEAN_IA`	Interarrival rate of new CS3 sessions: Default: `CS3_MEAN_IA = 274350`

Table 6: IMAP Client Type Load and Distributions

Each `CLIENT_TYPE_DISTRIBUTION` tuple defines the command sequence grouping. The first element is a list of Command Sequence numbers (1 == CS1, 34 == CS3+CS4). The second element is the percentage of overall load generator client threads that will implement each combination. The number of IMAP sessions varies as this matrix changes. Each load generator thread is assigned one specific combination.

The total number of IMAP sessions is related directly to the total number of active users, as distributed by CS1, CS2, CS4 and CS5 percentages. Only CS3 is not included in the total since its session length is determined by SUT response times and number of subscribed folders.

The effective Client Type distribution applied to the base UserCount totals to 136%.

CS1	CS2	CS3	CS4	CS5
53.92%	1.96%	40.20%	27.45%	12.75%

Table 7: Effective IMAP Client Type Distribution

5.3.5 *IMAP Benchmark's States*

The analysis process classified individual extracted IMAP sessions according to the rules in Table 4: IMAP Command Sequence Definition, and the actual IMAP4 command and parameter combination mapped to common states. The state-to-state transitions were then collated and the corresponding probabilities (represented as percentages) collected. The analysis process created a large number of states (234), and a wide variety of possible state transitions (from 1 to 24). However, further analysis reduced the large number of states to only 64 states, maximum. The restrictions include

- commands needed to establish one of the five Command Sequences
- commands present during the peak hour
- commands that represented a more than 5 percent of the total number
- commands between 1 and 5% that should incur a disproportional computing resource, such as SEARCH or moving messages between folders

Table 8 lists all derived IMAP command state names, their numeric state ID code, and which data source used it. Some states are variations of each other (same command but slight parameter variation) because of the four different IMAP client types. These different clients used these variants for the same purpose. The benchmark treats these as unique command states, based on Client Type affiliation.

Command states found in the Peak Hour data samples, are marked with an 'X' in the table grid.

Table 8: Detected IMAP4 Command-States

State Identfier	State Name	Mirapoint	Sun	University of Wollongang
1.	APPEND	X	X	X
2.	CHECK	X	X	X
3.	CLOSE			
4.	COPY_NUM_FOLDER			
5.	COPY_RANGE_FOLDER			
6.	CREATE			X
7.	DELETE			
8.	EXAMINE_FOLDER			
9.	EXAMINE_INBOX			
10.	EXAMINE_INBOXSENT			
11.	EXAMINE_SENT			
12.	EXAMINE_SENT_ITEMS			
13.	EXPUNGE	X	X	X
14.	FETCH_NUM			
15.	FETCH_NUM_BODYALL	X	X	
16.	FETCH_NUM_BODYPARTS			
17.	FETCH_NUM_BODYPEEK	X		
18.	FETCH_NUM_BODYPEEK_HEADER	X		
19.	FETCH_NUM_BODYPEEK_HEADERFIELDS			
20.	FETCH_NUM_BODYSTRUCTURE_FLAGS	X	X	
21.	FETCH_NUM_BODY_BODYALL_HEADERFIELDS			
22.	FETCH_NUM_BODY_HEADER			
23.	FETCH_NUM_ENVELOPE_BODYPEEK_HEADERFIELDS_BODYSTRUCTURE_FLAGS_INTERNALDATE_RFC822SIZE			
24.	FETCH_NUM_ENVELOPE_BODYPEEK_HEADERFIELDS_FLAGS_INTERNALDATE_RFC822SIZE_UID			
25.	FETCH_NUM_ENVELOPE_BODYPEEK_HEADERFIELDS_FLAGS_INTERNALDATE	X	X	

State Identfier	State Name	Mirapoint	Sun	University of Wollongang
	_RFC822SIZE_UID			
26.	FETCH_NUM_FLAGS	X		
27.	FETCH_NUM_FLAGS_BODYPEEK_ HEADERFIELDS_INTERNALDATE_RFC822SIZE			
28.	FETCH_NUM_FLAGS_BODYSTRUCTURE _ENVELOPE_INTERNALDATE_RFC822SIZE_UID			
29.	FETCH_NUM_RFC822HEADER			
30.	FETCH_NUM_RFC822TEXT			
31.	FETCH_NUM_UID	X		
32.	FETCH_NUM_UID_BODYPEEK_HEADERFIELDS _ENVELOPE_FLAGS_INTERNALDATE_RFC822SIZE	X		
33.	FETCH_RANGE_UID	X		
34.	FETCH_RANGE_BODYPEEK_HEADERFIELDS			
35.	FETCH_RANGE_ENVELOPE_BODYPEEK_ HEADERFIELDS_FLAGS_INTERNALDATE_ RFC822SIZE_UID			
36.	FETCH_RANGE_FLAGS_BODYPEEK _HEADERFIELDS _INTERNALDATE_RFC822SIZE			
37.	FETCH_RANGE_FLAGS_ BODYSTRUCTURE_ENVELOPE _INTERNALDATE_RFC822SIZE_UID	X		
38.	FETCH_RANGE_UID_BODYPEEK_HEADERFIELDS _ENVELOPE_FLAGS_INTERNALDATE_RFC822SIZE	X		
39.	FETCH_SERIES_ENVELOPE_ BODYPEEK_HEADERFIELDS_ FLAGS_INTERNALDATE_RFC822SIZE_UID			
40.	FETCH_SERIES_ENVELOPE_ BODYSTRUCTURE_INTERNALDATE_RFC822SIZE			
41.	FETCH_SERIES_FLAGS_BODYPEEK_HEADERFIELDS _INTERNALDATE_RFC822SIZE			
42.	FETCH_SERIES_UID			
43.	FETCH_UID			
44.	LIST	X	X	X
45.	LOGIN	X	X	X
46.	LOGOUT	X	X	X
47.	LSUB_NULL_FOLDER		X	X
48.	LSUB_NULL_PART			
49.	LSUB_NULL_SENT			
50.	LSUB_NULL_WILDCARD	X	X	X
51.	LSUB_WILDCARD_WILDCARD			
52.	NOOP	X	X	X
53.	RENAME_FOLDER_FOLDER			
54.	RENAME_INBOXINBOXSENT _INBOXTRASHINBOXSENT			
55.	SEARCH_ALL_DELETED	X		
56.	SEARCH_ALL_RANGE_CHARSET_RFCHEADER			
57.	SEARCH_ALL_RFCHEADER			
58.	SEARCH_ALL_UNDELETED_UNSEEN			
59.	SEARCH_DELETED	X		
60.	SEARCH_RFCHEADER			
61.	SEARCH_UNDELETED			
62.	SEARCH_UNSEEN	X		
63.	SELECT_			
64.	SELECT_FOLDER	X	X	
65.	SELECT_FOLDER_ITEMS			
66.	SELECT_INBOX	X	X	X
67.	SELECT_INBOXSENT	X		
68.	SELECT_INBOXSENT_ITEMS			
69.	SELECT_SENT			
70.	SELECT_SENT_ITEMS			
71.	STARTED			
72.	STATUS_FOLDER_ITEMS_MESSAGES_UNSEEN			
73.	STATUS_FOLDER_ITEMS_UNSEEN			
74.	STATUS_FOLDER_MESSAGES			

State Identfier	State Name	Mirapoint	Sun	University of Wollongang
75.	STATUS_FOLDER_MESSAGES_RECENT _UNSEEN_UIDVALIDITY_UIDNEXT			
76.	STATUS_FOLDER_MESSAGES_UNSEEN			
77.	STATUS_FOLDER_UIDNEXT			
78.	STATUS_FOLDER_UIDNEXT_ UIDVALIDITY_MESSAGES			
79.	STATUS_FOLDER_UNSEEN			
80.	STATUS_INBOXSENT_ITEMS_MESSAGES_UNSEEN			
81.	STATUS_INBOXSENT_ITEMS_UNSEEN			
82.	STATUS_INBOXSENT_UNSEEN			
83.	STATUS_INBOXSENT_MESSAGES_UNSEEN			
84.	STATUS_INBOX_MESSAGES_RECENT _UNSEEN_UIDVALIDITY_UIDNEXT			
85.	STATUS_INBOX_MESSAGES_UNSEEN			
86.	STATUS_INBOX_UIDNEXT			
87.	STATUS_INBOX_UIDNEXT_ UIDVALIDITY_MESSAGES			
88.	STATUS_INBOX_UNSEEN			
89.	STATUS_SENT_ITEMS_MESSAGES_UNSEEN			
90.	STATUS_SENT_ITEMS_UNSEEN			
91.	STATUS_SENT_MESSAGES_UNSEEN			
92.	STATUS_SENT_UNSEEN			
93.	STORE_NUM_SET_FLAGS_ANSWERED			
94.	STORE_NUM_SET_FLAGS_DELETED	X	X	
95.	STORE_NUM_SET_FLAGS_SEEN			
96.	STORE_NUM_UNSET_FLAGS_DELETED			
97.	STORE_NUM_UNSET_FLAGS_SEEN			
98.	STORE_RANGE_SET_FLAGS_DELETED			
99.	STORE_RANGE_SET_FLAGS_SEEN			
100.	STORE_SERIES_SET_FLAGS_DELETED			
101.	STORE_UNTILEND_SET_FLAGS_DELETED			
102.	STORE_UNTILEND_SET_FLAGS_SEEN			
103.	SUBSCRIBE_FOLDER			
104.	SUBSCRIBE_INBOXSENT			
105.	UID_COPY_NUM_FOLDER	X	X	X
106.	UID_COPY_NUM_INBOX			
107.	UID_COPY_NUM_INBOXSENT			
108.	UID_COPY_RANGE_FOLDER	X	X	X
109.	UID_COPY_RANGE_INBOX			
110.	UID_COPY_RANGE_INBOXSENT			
111.	UID_COPY_SERIES_FOLDER		X	X
112.	UID_FETCH_NUM_BODY			
113.	UID_FETCH_NUM_BODYALL	X	X	X
114.	UID_FETCH_NUM_BODYPARTS	X	X	X
115.	UID_FETCH_NUM_BODYPEEK			X
116.	UID_FETCH_NUM_BODYPEEKALL			X
117.	UID_FETCH_NUM_BODYPEEK_HEADER			X
118.	UID_FETCH_NUM_BODYPEEK_UID			
119.	UID_FETCH_NUM_BODYSTRUCTURE	X	X	X
120.	UID_FETCH_NUM_BODY _BODYMIMEALL _BODYMIMEPARTS_HEADER	X	X	X
121.	UID_FETCH_NUM_BODY_ BODYMIMEALL_HEADER	X	X	X
122.	UID_FETCH_NUM_BODY_HEADER			
123.	UID_FETCH_NUM_ENVELOPE			
124.	UID_FETCH_NUM_FLAGS			
125.	UID_FETCH_NUM_RFC822SIZE			X
126.	UID_FETCH_NUM_UID			
127.	UID_FETCH_NUM_UID_BODYPEEK_ FLAGS_INTERNALDATE			
128.	UID_FETCH_NUM_UID_BODYPEEK_ FLAGS_INTERNALDATE_RFC822SIZE			
129.	UID_FETCH_NUM_UID_BODYPEEK_ HEADERFIELDS_FLAGS_RFC822SIZE	X	X	X

State Identfier	State Name	Mirapoint	Sun	University of Wollongang
130.	UID_FETCH_NUM_UID_BODYPEEK_ HEADER_FLAGS_INTERNALDATE_RFC822SIZE			
131.	UID_FETCH_NUM_UID_BODYPEEK_RFC822SIZE	X		X
132.	UID_FETCH_NUM_UID_BODY_RFC822SIZE	X		X
133.	UID_FETCH_RANGE_UID_ BODYPEEK_FLAGS_INTERNALDATE			
134.	UID_FETCH_RANGE_UID_BODYPEEK _HEADERFIELDS_FLAGS_RFC822SIZE	X	X	X
135.	UID_FETCH_RANGE_UID_BODYPEEK _RFC822SIZE	X		
136.	UID_FETCH_RANGE_UID_ENVELOPE _FLAGS_INTERNALDATE_RFC822SIZE			
137.	UID_FETCH_RANGE_UID_FLAGS			
138.	UID_FETCH_RANGE_UID_RFC822SIZE _BODYPEEK_HEADERFIELDS			
139.	UID_FETCH_RANGE_UID_UID_BODYPEEK_ HEADER_HEADERFIELDS_FLAGS_ FLAGS_RFC822SIZE_RFC822SIZE_UID			
140.	UID_FETCH_SERIES_UID_ BODYPEEK_FLAGS_INTERNALDATE			
141.	UID_FETCH_SERIES_UID_ BODYPEEK_HEADERFIELDS_FLAGS_RFC822SIZE			
142.	UID_FETCH_SERIES_UID_ BODYPEEK_RFC822SIZE			X
143.	UID_FETCH_UID_BODYPEEK_ HEADERFIELDS_FLAGS_RFC822SIZE			
144.	UID_FETCH_UID_BODYPEEK_ HEADER_FLAGS_INTERNALDATE_RFC822SIZE			
145.	UID_FETCH_UNTILEND_BODYPEEK_ HEADERFIELDS_ENVELOPE_FLAGS _INTERNALDATE_RFC822SIZE_UID			
146.	UID_FETCH_UNTILEND_ENVELOPE _FLAGS_INTERNALDATE_RFC822SIZE_UID			
147.	UID_FETCH_UNTILEND_FLAGS	X	X	X
148.	UID_FETCH_UNTILEND_UID_BODYPEEK _HEADERFIELDS_FLAGS_RFC822SIZE			X
149.	UID_FETCH_UNTILEND_UID_BODYPEEK _HEADER_FLAGS_INTERNALDATE_RFC822SIZE			
150.	UID_FETCH_UNTILEND_UID_FLAGS			
151.	UID_FETCH_UNTILEND_UID_FLAGS _INTERNALDATE_RFC822HEADER_RFC822SIZE			
152.	UID_SEARCH_ANSWERED			
153.	UID_SEARCH_DELETED			X
154.	UID_SEARCH_FLAGGED			
155.	UID_SEARCH_HEADER_ QUESTION_RFCHEADER_UNDELETED			
156.	UID_SEARCH_HEADER_ RFCHEADER_UNDELETED			
157.	UID_SEARCH_HEADER_UNDELETED	X	X	
158.	UID_SEARCH_KEYWORD			
159.	UID_SEARCH_NOTDELETED_UID_UNTILEND			
160.	UID_SEARCH_RFCHEADER_UNDELETED			X
161.	UID_SEARCH_SEEN			
162.	UID_SEARCH_SINCE			
163.	UID_SEARCH_UID_NUM			
164.	UID_SEARCH_UID_NUM_NOTDELETED			
165.	UID_SEARCH_UID_RANGE			
166.	UID_SEARCH_UID_RANGE_NOTDELETED			
167.	UID_SEARCH_UID_UNTILEND_ UNDELETED_UNDRAFT_UNSEEN			
168.	UID_SEARCH_UID_UNTILEND _UNDELETED_UNSEEN			
169.	UID_SEARCH_UNDELETED			
170.	UID_SEARCH_UNDELETED_UNSEEN			
171.	UID_SEARCH_UNSEEN			X
172.	UID_SEARCH_UNTILEND_			

State Identfier	State Name	Mirapoint	Sun	University of Wollongang
173.	UID_STORE_NUM_SET_FLAGS _ANSWERED	X	X	X
174.	UID_STORE_NUM_SET_FLAGS_ ANSWERED_DELETED_SEEN			
175.	UID_STORE_NUM_SET_FLAGS_ ANSWERED_SEEN			
176.	UID_STORE_NUM_SET_FLAGS_DELETED	X	X	X
177.	UID_STORE_NUM_SET_FLAGS_ DELETED_SEEN			X
178.	UID_STORE_NUM_SET_FLAGS_FLAGGED		X	
179.	UID_STORE_NUM_SET_FLAGS_SEEN	X	X	X
180.	UID_STORE_NUM_SET_FLAGS_SEEN _ANSWERED			
181.	UID_STORE_NUM_SET_FLAGS_SEEN _DELETED		X	
182.	UID_STORE_NUM_UNSET_FLAGS_			
183.	UID_STORE_NUM_UNSET_FLAGS_ANSWERED			X
184.	UID_STORE_NUM_UNSET_FLAGS_DELETED		X	
185.	UID_STORE_NUM_UNSET_FLAGS_FLAGGED		X	
186.	UID_STORE_NUM_UNSET_FLAGS_ FLAGGED_ANSWERED			
187.	UID_STORE_NUM_UNSET_FLAGS_FLAGGED _FORWARDED_MDNSENT_DELETED_DRAFT			
188.	UID_STORE_NUM_UNSET_FLAGS_SEEN		X	X
189.	UID_STORE_NUM_UNSET_FLAGS_SEEN _ANSWERED			
190.	UID_STORE_NUM_UNSET_FLAGS_SEEN _ANSWERED_DELETED			
191.	UID_STORE_NUM_UNSET_FLAGS_SEEN _ANSWERED_DELETED_DRAFT_FLAGGED			
192.	UID_STORE_NUM_UNSET_FLAGS_SEEN_ ANSWERED_DELETED_FLAGGED			
193.	UID_STORE_NUM_UNSET_FLAGS_ SEEN_ANSWERED_FLAGGED			
194.	UID_STORE_NUM_UNSET_FLAGS_SEEN_DELETED			
195.	UID_STORE_NUM_UNSET_FLAGS_SEEN_FLAGGED			
196.	UID_STORE_NUM_UNSET_FLAGS_ SEEN_FORWARDED_MDNSENT_ ANSWERED_DELETED_DRAFT_FLAGGED			
197.	UID_STORE_NUM_UNSET_FLAGS _SEEN_FORWARDED_MDNSENT_ DELETED_DRAFT_FLAGGED			
198.	UID_STORE_NUM_UNSET_FLAGS_ SEEN_MDNSENT_ANSWERED_ DELETED_DRAFT_FLAGGED			
199.	UID_STORE_RANGE_SET_FLAGS_ANSWERED			
200.	UID_STORE_RANGE_SET_FLAGS_DELETED	X	X	X
201.	UID_STORE_RANGE_SET_FLAGS_DELETED_SEEN			
202.	UID_STORE_RANGE_SET_FLAGS_SEEN			
203.	UID_STORE_RANGE_SET_FLAGS_SEEN_DELETED			
204.	UID_STORE_RANGE_UNSET_FLAGS_1006190098397113			
205.	UID_STORE_RANGE_UNSET_FLAGS_ANSWERED_ FORWARDED_MDNSENT_DELETED_DRAFT_FLAGGED			
206.	UID_STORE_RANGE_UNSET_FLAGS_DELETED			
207.	UID_STORE_RANGE_UNSET_FLAGS_SEEN			
208.	UID_STORE_RANGE_UNSET_FLAGS_SEEN_ FORWARDED_MDNSENT_ANSWERED_ DELETED_DRAFT_FLAGGED			
209.	UID_STORE_SERIES_SET_FLAGS_DELETED		X	X
210.	UID_STORE_SERIES_SET_FLAGS_DELETED_SEEN			
211.	UID_STORE_SERIES_SET_FLAGS_SEEN			
212.	UID_STORE_SERIES_UNSET_FLAGS_SEEN _FORWARDED_MDNSENT_ANSWERED _DELETED_DRAFT_FLAGGED			
213.	UID_STORE_UNSET_FLAGS_SEEN			
214.	UNSUBSCRIBE_FOLDER			

State Identfier	State Name	Mirapoint	Sun	University of Wollongang
215.	SEARCH_ALL_CALL_INFORMATION		X	
216.	UID_COPY_NUM_		X	
217.	UID_COPY_NUM_TRASH		X	X
218.	UID_COPY_RANGE_TRASH		X	
219.	UID_COPY_SERIES_TRASH		X	
220.	UID_FETCH_NUM_BODYPEEK_RFC822SIZE_UID		X	
221.	UID_FETCH_NUM_BODY_RFC822SIZE_UID		X	
222.	UID_FETCH_NUM_UID_BODYPEEK_HEADER_FLAGS _RFC822SIZE		X	
223.	UID_FETCH_RANGE_UID_BODYPEEK_HEADER_FLAG S _RFC822SIZE		X	
224.	LSUB_			X
225.	LSUB_.			X
226.	SUBSCRIBE_TRASH			X
227.	UID_COPY_NUM_.			X
228.	UID_COPY_RANGE_.			X
229.	UID_COPY_SERIES_.			X
230.	UID_FETCH_NUM_BODYMIMEALL			X
231.	UID_FETCH_NUM_UID_BODYSTRUCTURE			X
232.	UID_FETCH_RANGE_BODYPEEK_HEADERFIELDS			X
233.	UID_FETCH_UNTILEND_FLAGS_RFC822SIZE			X
234.	SESSION_START	X	X	X

5.3.6 SPECmail2008 State Engine Components

The benchmark's state engine is driven by a set of internally defined tables, found in the CS1State.java, CS2State.java, CS3State.java, CS4State.java, and CS5State.java files.

Benchmark Global Table	**Definition and Value**
`TOSTATE[][]`	An array of State Identifiers. Each row in the array corresponds to the integer State ID. Each column represents one of the possible "next" states.

```
Example:

CS1STATE::TOSTATE[][] = {
{ 0 }, // 0 Place holder
{ 1, 23, 25, 27, 32, 33, 56 }, // 1
{ 12, 23, 27, 54, 60 }, // 2
...
{ 12, 14, 15, 25, 27, 48, 51, 53, 54 }, // 5
{ 1, 2, 6, 7, 9, 11, 15, 27, 36, 37 }, // 6
{ 6, 7, 11, 37 }, // 7
...
{ 2, 25, 27, 40, 50, 56, 57, 59, 61 }, // 59
{ 27, 40, 47, 50, 54, 57 }, // 60
{ 50, 59, 61 }, // 61
{ 5, 27, 40, 47, 50, 56 }, // 62
{ 27 }, // 63
{ 5, 42, 47, 50 } // 64
```

`TOSTATEPERCENT[][]`	Defines the probability (column) that each state present in CS2 (row) moves to the next state, as defined in the corresponding TOSTATE array.

```
Example:
CS2STATE::TOSTATEPERCENT[][] = {
{ 0.0000 }, // 0 Place holder
{ 0.2000, 0.2000, 0.2000, 0.4000 }, // 1
{ 0.0465, 0.6744, 0.1628, 0.0698 }, // 2
{ 1.0000 }, // 3
{ 1.0000 }, // 4
{ 0.7500, 0.2500 }, // 5
{ 1.0000 }, // 6
{ 0.0132, 0.0132, 0.1711, 0.8026 }, // 7
{ 0.0030, 0.0030, 0.0030, 0.1875, 0.8036 }, // 8
{ 1.0000 } // 9
```

`toStateCount[][]`
`toStateAllowed[][]`

These two pre-sized tables to hold occurrence counts and transition permissions. These two table ensure that every state (column) transition has been invoked, in the proper proportion. This enforces compliance with the TOSTATEPERCENT table, in case the “random” function is not very random.

`TOSTATEIARATE[][]`

This array defines the State-to-State Inter-Arrival wait time computation type (LOGNORM, in most cases) and the derived from the IMAP session data samples. As with the TOSTATE and TOSTATEPERCENT, each row represents a specific IMAP4 command and parameter combination. Each column represents the minimum wait time before moving to the corresponding “next” state.

```
Example:
CS4STATE:: TOSTATEIARATE[][]= {
{ "Place holder" }, // 0 Place holder
{ "LOGNORM:349.6307", "LOGNORM: 3386.3247",
"LOGNORM:395.4172", ... }, // 1
{ "LOGNORM:0.0998" }, // 2
{ "LOGNORM:0.0718" }, // 3
{ "LOGNORM:0.0988" }, // 4
{ "LOGNORM:0.1369", "LOGNORM
{ "LOGNORM:459.2843", "LOGNO
{ "LOGNORM:0.0546", "LOGNORM
{ "LOGNORM:0.0000" }, // 8
{ "LOGNORM:0.0032", "LOGNORM:0027", "LOGNORM:0.0079" },
{ "LOGNORM:428.7473", "LOGNO
{ "LOGNORM:1.4748" }, // 11
{ "LOGNORM:1.1354" }, // 12
{ "LOGNORM:0.0000" }, // 13
{ "LOGNORM:0.0153" }, // 14
{ "LOGNORM:0.1084" }, // 15
{ "LOGNORM:0.0932" }, // 16
...
```

`STATEIDTOMETACOMMAND[]`

The actual State Identification strings to be used for internal to external reports, such as debug statements and results.

```
Example:
CS4STATE::STATEIDTOMETACOMMAND[]={
"Place holer", // 0 Place holder
"APPEND", //1
"CHECK", //2
"CLOSE", //3
```

```
"CREATE", //4
"EXPUNGE", //5
"LIST", //6
"LOGIN", //7
"LOGOUT", //8
"NOOP", //9
"SELECT_FOLDER", //10
"SELECT_INBOX", //11
"SELECT_INBOXSENT", //12
"SESSION_START", //13
"SUBSCRIBE_FOLDER", //14
"UID_COPY_NUM_FOLDER", //15
"UID_COPY_RANGE_FOLDER", //16
"UID_FETCH_NUM_BODYALL", //17
...
```

Table 9: IMAP State Transition Tables and Probabilities

6. Benchmark Reportable Parameters:

The following table shows IMAP4 commands deemed critical enough to consider reportable, and whether to consider the corresponding Quality-of-Service (QoS) value.

Table 10: Reportable IMAP Commands

IMAP Command	Report for Workload	Report for QOS
LOGIN	Yes	Yes
LOGOUT	Yes	Yes
NOOP	Yes	No
SEARCH	Yes	No
UIDSEARCH	Yes	No
APPEND	Yes	
EXPUNGE	Yes	
STORE	Yes	
FETCH (Single) Body	Yes	Yes
FETCH (Many) Body	Yes	
FETCH (Single) Meta	Yes	
FETCH (Many) Meta	Yes	
FETCH (Single) Header	Yes	
FETCH (Many) Header	Yes	
UIDFETCH (Single) Body	Yes	Yes
UIDFETCH (Many) Body	Yes	
UIDFETCH (Single) Meta	Yes	
UIDFETCH (Many) Meta	Yes	
UIDFETCH (Single) Header	Yes	
UIDFETCH (Many) Header	Yes	
COPY (Single)	Yes	
COPY (Many)	Yes	
UIDCOPY (Single)	Yes	
UIDCOPY (Many)	Yes	
CHECK	Yes	
CREATE	Yes	Yes
DELETE	Yes	Yes

EXAMINE	Yes	
LIST	Yes	
LSUB	Yes	
RENAME	Yes	Yes
SELECT	Yes	
STATUS	Yes	Yes
SUBSCRIBE	Yes	Yes
UNSUBSCRIBE	Yes	Yes

7. References:

7.1 Relevant RFCs (see www.ietf.org):

2045 – Part 1: Format of Internet Message Bodies
2046 – Part 2: Media types
2047 – Part 3: Header and Body Extensions for non-ASCII Text, non Textual message parts and multi-part messages
2048 – Part 4: Registrations
2049 – Part 5: Conformance Criteria and Examples
2231 – Extension to specify the language to display the part, parameter values in other (non US-ASCII) character sets, and continuation mechanism for long parameter values.
2646 – Update to RFC 2046 to define variations of supported "Plain/Text" content types to incorporate legacy plain text and flow line control.