Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 1, 2009

Stone, Ed.
Six Apart, Ltd.
Norbye, Ed.
Sun Microsystems, INC

August 28, 2008

Memcache Binary Protocol
draft-stone-memcache-binary-01

Status of this Memo

This Internet-Draft will expire on March 1, 2009.

Abstract

This memo explains the memcache binary protocol for informational purposes.

Memcache is a high performance key-value cache. It is intentionally a dumb cache, optimized for speed only. Applications using memcache do not rely on it for data -- a persistent database with guaranteed reliability is strongly recommended -- but applications can run much faster when cached data is available in memcache.

Table of Contents

TOC

1. Introduction

Memcache is a high performance key-value cache. It is intentionally a dumb cache, optimized for speed only. Applications using memcache should not rely on it for data -- a persistent database with guaranteed reliability is strongly recommended -- but applications can run much faster when cached data is available in memcache.

Memcache was originally written to make LiveJournal faster. It now powers all of the fastest web sites that you love.

TOC

1.1. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in

---

TOC

2. Packet Structure

General format of a packet:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0/ HEADER                                                        /
    /                                                               /
    /                                                               /
    /                                                               /
    +---------------+---------------+---------------+---------------+
  24/ COMMAND-SPECIFIC EXTRAS (as needed)                           /
   +/  (note length in the extras length header field)              /
```

```
     +---------------+---------------+---------------+---------------+
    m/ Key (as needed)                                               /
    +/  (note length in key length header field)                     /
     +---------------+---------------+---------------+---------------+
    n/ Value (as needed)                                             /
    +/  (note length is total body length header field, minus        /
    +/   sum of the extras and key length body fields)               /
     +---------------+---------------+---------------+---------------+
     Total 24 bytes
```

Request header:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| Magic         | Opcode        | Key length                    |
    +---------------+---------------+---------------+---------------+
   4| Extras length | Data type     | Reserved                      |
    +---------------+---------------+---------------+---------------+
   8| Total body length                                             |
    +---------------+---------------+---------------+---------------+
  12| Opaque                                                        |
    +---------------+---------------+---------------+---------------+
  16| CAS                                                           |
    |                                                               |
    +---------------+---------------+---------------+---------------+
    Total 24 bytes
```

Response header:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| Magic         | Opcode        | Key Length                    |
    +---------------+---------------+---------------+---------------+
   4| Extras length | Data type     | Status                        |
    +---------------+---------------+---------------+---------------+
   8| Total body length                                             |
    +---------------+---------------+---------------+---------------+
  12| Opaque                                                        |
    +---------------+---------------+---------------+---------------+
  16| CAS                                                           |
    |                                                               |
    +---------------+---------------+---------------+---------------+
    Total 24 bytes
```

Header fields:

Magic
    Magic number.
Opcode
    Command code.
Key length
    Length in bytes of the text key that follows the command extras.
Status
    Status of the response (non-zero on error).
Extras length
    Length in bytes of the command extras.
Data type
    Reserved for future use (Sean is using this soon).

Reserved
    Really reserved for future use (up for grabs).
Total body length
    Length in bytes of extra + key + value.
Opaque
    Will be copied back to you in the response.
CAS
    Data version check.

TOC

3. Defined Values

TOC

3.1. Magic Byte

0x80
    Request packet for this protocol version
0x81
    Response packet for this protocol version

Magic byte / version. For each version of the protocol, we'll use a different request/response value pair. This is useful for protocol analyzers to distinguish the nature of the packet from the direction which it is moving. Note, it is common to run a memcached instance on a host that also runs an application server. Such a host will both send and receive memcache packets.

The version should hopefully correspond only to different meanings of the command byte. In an ideal world, we will not change the header format. As reserved bytes are given defined meaning, the protocol version / magic byte values should be incremented.

Traffic analysis tools are encouraged to identify memcache packets and provide detailed interpretation if the magic bytes are recognized and otherwise to provide a generic breakdown of the packet. Note, that the key and value positions can always be identified even if the magic byte or command opcode are not recognized.

TOC

3.2. Response Status

Possible values of this two-byte field:

0x0000
    No error
0x0001
    Key not found
0x0002
    Key exists
0x0003
    Value too large
0x0004
    Invalid arguments
0x0005
    Item not stored
0x0006
    Incr/Decr on non-numeric value.
0x0081
    Unknown command
0x0082

Out of memory

TOC

3.3. Command Opcodes

Possible values of the one-byte field:

0x00	Get
0x01	Set
0x02	Add
0x03	Replace
0x04	Delete
0x05	Increment
0x06	Decrement
0x07	Quit
0x08	Flush
0x09	GetQ
0x0A	No-op
0x0B	Version
0x0C	GetK
0x0D	GetKQ
0x0E	Append
0x0F	Prepend
0x10	Stat
0x11	SetQ
0x12	AddQ
0x13	ReplaceQ
0x14	DeleteQ
0x15	IncrementQ
0x16	DecrementQ
0x17	QuitQ
0x18	FlushQ
0x19	AppendQ
0x1A	

PrependQ

As a convention all of the commands ending with "Q" for Quiet. A quiet version of a command will omit responses that are considered uninteresting. Whether a given response is interesting is dependent upon the command. See the descriptions of the **set commands** and **set commands** for examples of commands that include quiet variants.

TOC

3.4. Data Types

Possible values of the one-byte field:

0x00
Raw bytes

TOC

4. Commands

TOC

4.1. Introduction

All communication is initiated by a request from the client, and the server will respond to each request with zero or multiple packets for each request. If the status code of a response packet is non-nil, the body of the packet will contain a textual error message. If the status code is nil, the command opcode will define the layout of the body of the message.

TOC

4.1.1. Example

The following figure illustrates the packet layout for a packet with an error message.

Packet layout:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x81          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x01          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x09          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0x4e ('N')    | 0x6f ('o')    | 0x74 ('t')    | 0x20 (' ')    |
    +---------------+---------------+---------------+---------------+
  28| 0x66 ('f')    | 0x6f ('o')    | 0x75 ('u')    | 0x6e ('n')    |
    +---------------+---------------+---------------+---------------+
  32| 0x64 ('d')    |
    +---------------+
    Total 33 bytes (24 byte header, and 9 bytes value)
```

```
Field        (offset) (value)
Magic        (0)    : 0x81
Opcode       (1)    : 0x00
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Status       (6,7)  : 0x0001
Total body   (8-11) : 0x00000009
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key                 : None
Value        (24-32): The textual string "Not found"
```

TOC

4.2. Get, Get Quietly, Get Key, Get Key Quietly

Request:

MUST NOT have extras.

MUST have key.

MUST NOT have value.

Response (if found):

MUST have extras.

MAY have key.

MAY have value.

- 4 byte flags

Extra data for the get commands:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| Flags                                                         |
    +---------------+---------------+---------------+---------------+

    Total 4 bytes
```

The get command gets a single key. The getq command is both mum on cache miss and quiet, holding its response until a non-quiet command is issued. Getk and getkq differs from get and getq by adding the key into the response packet.

You're not guaranteed a response to a getq/getkq cache hit until you send a non-getq/getkq command later, which uncorks the server and bundles up IOs to send to the client in one go.

Clients should implement multi-get (still important for reducing network roundtrips!) as n pipelined requests, the first n-1 being getq/getkq, the last being a regular get/getk. That way you're guaranteed to get a response, and you know when the server's done. You can also do the naive thing and send n pipelined get/getks, but then you could potentially get back a lot of "NOT_FOUND" error code packets. Alternatively, you can send 'n' getq/getkqs, followed by a 'noop' command.

4.2.1. Example

TOC

To request the data associated with the key "Hello" the following fields must be specified in the packet.

get request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x00          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0x48 ('H')    | 0x65 ('e')    | 0x6c ('l')    | 0x6c ('l')    |
    +---------------+---------------+---------------+---------------+
  28| 0x6f ('o')    |
    +---------------+

    Total 29 bytes (24 byte header, and 5 bytes key)

Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x00
Key length   (2,3)  : 0x0005
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000005
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key          (24-29): The textual string: "Hello"
Value               : None
```

If the item exist on the server the following packet is returned, otherwise a packet with status code != 0 will be returned (see **Introduction**)

get/getq response:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x81          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x04          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x09          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x01          |
     +---------------+---------------+---------------+---------------+
   24| 0xde          | 0xad          | 0xbe          | 0xef          |
     +---------------+---------------+---------------+---------------+
   28| 0x57 ('W')    | 0x6f ('o')    | 0x72 ('r')    | 0x6c ('l')    |
     +---------------+---------------+---------------+---------------+
   32| 0x64 ('d')    |
     +---------------+

     Total 33 bytes (24 byte header, 4 byte extras and 5 byte value)

Field        (offset) (value)
Magic        (0)    : 0x81
Opcode       (1)    : 0x00
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x04
Data type    (5)    : 0x00
Status       (6,7)  : 0x0000
Total body   (8-11) : 0x00000009
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000001
Extras              :
  Flags      (24-27): 0xdeadbeef
Key                 : None
Value        (28-32): The textual string "World"
```

getk/getkq response:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x81          | 0x00          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
   4| 0x04          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x09          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x01          |
    +---------------+---------------+---------------+---------------+
  24| 0xde          | 0xad          | 0xbe          | 0xef          |
    +---------------+---------------+---------------+---------------+
  28| 0x48 ('H')    | 0x65 ('e')    | 0x6c ('l')    | 0x6c ('l')    |
    +---------------+---------------+---------------+---------------+
  32| 0x6f ('o')    | 0x57 ('W')    | 0x6f ('o')    | 0x72 ('r')    |
    +---------------+---------------+---------------+---------------+
  36| 0x6c ('l')    | 0x64 ('d')    |
    +---------------+---------------+

    Total 38 bytes (24 byte header, 4 byte extras, 5 byte key
                    and 5 byte value)

Field        (offset) (value)
Magic        (0)    : 0x81
Opcode       (1)    : 0x00
Key length   (2,3)  : 0x0005
Extra length (4)    : 0x04
Data type    (5)    : 0x00
Status       (6,7)  : 0x0000
Total body   (8-11) : 0x00000009
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000001
Extras              :
  Flags      (24-27): 0xdeadbeef
Key          (28-32): The textual string: "Hello"
Value        (33-37): The textual string: "World"
```

TOC

4.3. Set, Add, Replace

MUST have extras.

MUST have key.

MUST have value.

- 4 byte flags
- 4 byte expiration time

Extra data for set/add/replace:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| Flags                                                         |
    +---------------+---------------+---------------+---------------+
   4| Expiration                                                    |
    +---------------+---------------+---------------+---------------+
    Total 8 bytes
```

If the Data Version Check (CAS) is nonzero, the requested operation MUST only succeed if the item exists and has a CAS value identical to the provided value.

Add MUST fail if the item already exist.

Replace MUST fail if the item doesn't exist.

Set should store the data unconditionally if the item exists or not.

Quiet mutations only return responses on failure. Success is considered the general case and is suppressed when in quiet mode, but errors should not be allowed to go unnoticed.

TOC

4.3.1. Example

The following figure shows an add-command for

Key: "Hello"

Value: "World"

Flags: 0xdeadbeef

Expiry: in two hours

Add request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x02          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
   4| 0x08          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x12          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0xde          | 0xad          | 0xbe          | 0xef          |
    +---------------+---------------+---------------+---------------+
  28| 0x00          | 0x00          | 0x0e          | 0x10          |
    +---------------+---------------+---------------+---------------+
  32| 0x48 ('H')    | 0x65 ('e')    | 0x6c ('l')    | 0x6c ('l')    |
    +---------------+---------------+---------------+---------------+
  36| 0x6f ('o')    | 0x57 ('W')    | 0x6f ('o')    | 0x72 ('r')    |
    +---------------+---------------+---------------+---------------+
  40| 0x6c ('l')    | 0x64 ('d')    |
    +---------------+---------------+

    Total 42 bytes (24 byte header, 8 byte extras, 5 byte key and
                    5 byte value)

Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x02
Key length   (2,3)  : 0x0005
Extra length (4)    : 0x08
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000012
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              :
  Flags      (24-27): 0xdeadbeef
  Expiry     (28-31): 0x00000e10
Key          (32-36): The textual string "Hello"
Value        (37-41): The textual string "World"
```

The response-packet contains no extra data, and the result of the operation is signaled through the status code. If the command succeeds, the CAS value for the item is returned in the CAS-field of the packet.

Successful add response:

```
     Byte/     0       |       1       |       2       |       3       |
        /              |               |               |               |
       |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
       +---------------+---------------+---------------+---------------+
      0| 0x81          | 0x02          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
      4| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
      8| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     12| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     16| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     20| 0x00          | 0x00          | 0x00          | 0x01          |
       +---------------+---------------+---------------+---------------+

       Total 24 bytes
```

```
Field        (offset) (value)
Magic        (0)    : 0x81
Opcode       (1)    : 0x02
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Status       (6,7)  : 0x0000
Total body   (8-11) : 0x00000000
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000001
Extras              : None
Key                 : None
Value               : None
```

TOC

4.4. Delete

MUST NOT have extras.

MUST have key.

MUST NOT have value.

Delete the item with the specific key.

TOC

4.4.1. Example

The following figure shows a delete message for the item "Hello".

Delete request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x04          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0x48 ('H')    | 0x65 ('e')    | 0x6c ('l')    | 0x6c ('l')    |
    +---------------+---------------+---------------+---------------+
  28| 0x6f ('o')    |
    +---------------+

    Total 29 bytes (24 byte header, 5 byte value)
```

```
Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x04
Key length   (2,3)  : 0x0005
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000005
```

```
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key                 : The textual string "Hello"
Value               : None
```

The response-packet contains no extra data, and the result of the operation is signaled through the status code.

TOC

4.5. Increment, Decrement

MUST have extras.

MUST have key.

MUST NOT have value.

- 8 byte value to add / subtract
- 8 byte initial value (unsigned)
- 4 byte expiration time

Extra data for incr/decr:

```
Byte/     0       |       1       |       2       |       3       |
   /              |               |               |               |
  |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
  +---------------+---------------+---------------+---------------+
 0| Amount to add                                                 |
  |                                                               |
  +---------------+---------------+---------------+---------------+
 8| Initial value                                                 |
  |                                                               |
  +---------------+---------------+---------------+---------------+
16| Expiration                                                    |
  +---------------+---------------+---------------+---------------+
  Total 20 bytes
```

These commands will either add or remove the specified amount to the requested counter.

If the counter does not exist, one of two things may happen:

1. If the expiration value is all one-bits (0xffffffff), the operation will fail with NOT_FOUND.
2. For all other expiration values, the operation will succeed by seeding the value for this key with the provided initial value to expire with the provided expiration time. The flags will be set to zero.

incr/decr response body:

```
Byte/     0       |       1       |       2       |       3       |
   /              |               |               |               |
  |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
  +---------------+---------------+---------------+---------------+
 0| 64-bit unsigned response.                                     |
  |                                                               |
  +---------------+---------------+---------------+---------------+
  Total 8 bytes
```

4.5.1. Example

TOC

The following figure shows an incr-command for

Key: "counter"

Delta: 0x01

Initial: 0x00

Expiry: in two hours

Increment request:

```
     Byte/     0       |       1       |       2       |       3       |
        /              |               |               |               |
       |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
       +---------------+---------------+---------------+---------------+
      0| 0x80          | 0x05          | 0x00          | 0x07          |
       +---------------+---------------+---------------+---------------+
      4| 0x14          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
      8| 0x00          | 0x00          | 0x00          | 0x1b          |
       +---------------+---------------+---------------+---------------+
     12| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     16| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     20| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     24| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     28| 0x00          | 0x00          | 0x00          | 0x01          |
       +---------------+---------------+---------------+---------------+
     32| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     36| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     40| 0x00          | 0x00          | 0x0e          | 0x10          |
       +---------------+---------------+---------------+---------------+
     44| 0x63 ('c')    | 0x6f ('o')    | 0x75 ('u')    | 0x6e ('n')    |
       +---------------+---------------+---------------+---------------+
     48| 0x74 ('t')    | 0x65 ('e')    | 0x72 ('r')    |
       +---------------+---------------+---------------+
       Total 51 bytes (24 byte header, 20 byte extras, 7 byte key)
```

```
   Field        (offset) (value)
   Magic        (0)    : 0x80
   Opcode       (1)    : 0x05
   Key length   (2,3)  : 0x0007
   Extra length (4)    : 0x14
   Data type    (5)    : 0x00
   Reserved     (6,7)  : 0x0000
   Total body   (8-11) : 0x0000001b
   Opaque       (12-15): 0x00000000
   CAS          (16-23): 0x0000000000000000
   Extras              :
     delta      (24-31): 0x0000000000000001
     initial    (32-39): 0x0000000000000000
     exipration (40-43): 0x00000e10
   Key                 : Textual string "counter"
   Value               : None
```

If the key doesn't exist, the server will respond with the initial value. If not the incremented value will be returned. Let's assume that the key didn't exist, so the initial value is returned.

Increment response:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x81          | 0x05          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x08          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
  24| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  28| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
    Total 32 bytes (24 byte header, 8 byte value)
```

```
Field        (offset) (value)
Magic        (0)    : 0x81
Opcode       (1)    : 0x05
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Status       (6,7)  : 0x0000
Total body   (8-11) : 0x00000008
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000005
Extras              : None
Key                 : None
Value               : 0x0000000000000000
```

TOC

4.6. quit

MUST NOT have extras.

MUST NOT have key.

MUST NOT have value.

Close the connection to the server.

TOC

4.6.1. Example

Quit request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x07          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
    Total 24 bytes
```

```
Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x07
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000000
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key                 : None
Value               : None
```

The response-packet contains no extra data, and the result of the operation is signaled through the status code. The server will then close the connection.

4.7. Flush

TOC

MAY have extras.

MUST NOT have key.

MUST NOT have value.

- 4 byte expiration time

Extra data for flush:

```
Byte/     0       |       1       |       2       |       3       |
   /              |               |               |               |
  |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
  +---------------+---------------+---------------+---------------+
 0| Expiration                                                    |
  +---------------+---------------+---------------+---------------+
Total 4 bytes
```

Flush the items in the cache now or some time in the future as specified by the expiration field. See the documentation of the textual protocol for the full description on how to specify the expiration time.

4.7.1. Example

TOC

To flush the cache (delete all items) in two hours, the set the following values in the request

Flush request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x08          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x04          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x04          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0x00          | 0x00          | 0x0e          | 0x10          |
    +---------------+---------------+---------------+---------------+
    Total 28 bytes (24 byte header, 4 byte body)

Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x08
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x04
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000004
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              :
   Expiry    (24-27): 0x000e10
Key                 : None
Value               : None
```

The response-packet contains no extra data, and the result of the operation is signaled through the status code.

TOC

4.8. noop

MUST NOT have extras.

MUST NOT have key.

MUST NOT have value.

Used as a keep alive. Flushes outstanding getq/getkq's.

TOC

4.8.1. Example

Noop request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x0a          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
    Total 24 bytes
```

```
Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x0a
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000000
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key                 : None
Value               : None
```

The response-packet contains no extra data, and the result of the operation is signaled through the status code.

4.9. version

TOC

MUST NOT have extras.

MUST NOT have key.

MUST NOT have value.

Request the server version.

The server responds with a packet containing the version string in the body with the following format: "x.y.z"

4.9.1. Example

TOC

Version request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x0b          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
    Total 24 bytes
```

```
Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x0b
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000000
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
```

Version response:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x81          | 0x0b          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0x31 ('1')    | 0x2e ('.')    | 0x33 ('3')    | 0x2e ('.')    |
    +---------------+---------------+---------------+---------------+
  28| 0x31 ('1')    |
    +---------------+
    Total 29 bytes (24 byte header, 5 byte body)

Field        (offset) (value)
Magic        (0)    : 0x81
Opcode       (1)    : 0x0b
Key length   (2,3)  : 0x0000
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Status       (6,7)  : 0x0000
Total body   (8-11) : 0x00000005
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key                 : None
Value               : Textual string "1.3.1"
```

TOC

4.10. Append, Prepend

MUST NOT have extras.

MUST have key.

MUST have value.

These commands will either append or prepend the specified value to the requested key.

4.10.1. Example

TOC

The following example appends '!' to the 'Hello' key.

Append request:

```
  Byte/     0       |       1       |       2       |       3       |
     /              |               |               |               |
    |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
    +---------------+---------------+---------------+---------------+
   0| 0x80          | 0x0e          | 0x00          | 0x05          |
    +---------------+---------------+---------------+---------------+
   4| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
   8| 0x00          | 0x00          | 0x00          | 0x06          |
    +---------------+---------------+---------------+---------------+
  12| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  16| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  20| 0x00          | 0x00          | 0x00          | 0x00          |
    +---------------+---------------+---------------+---------------+
  24| 0x48 ('H')    | 0x65 ('e')    | 0x6c ('l')    | 0x6c ('l')    |
    +---------------+---------------+---------------+---------------+
  28| 0x6f ('o')    | 0x21 ('!')    |
    +---------------+---------------+
    Total 30 bytes (24 byte header, 5 byte key, 1 byte value)

Field        (offset) (value)
Magic        (0)    : 0x80
Opcode       (1)    : 0x0e
Key length   (2,3)  : 0x0005
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Reserved     (6,7)  : 0x0000
Total body   (8-11) : 0x00000006
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Extras              : None
Key          (24-28): The textual string "Hello"
Value        (29)   : "!"
```

The response-packet contains no extra data, and the result of the operation is signaled through the status code.

4.11. Stat

TOC

MUST NOT have extras.

MAY have key.

MUST NOT have value.

Request server statistics. Without a key specified the server will respond with a "default" set of statistics information. Each piece of statistical information is returned in its own packet (key contains the name of the statistical item and the body contains the value in ASCII format). The sequence of return packets is terminated with a packet that contains no key and no value.

4.11.1. Example

TOC

The following example requests all statistics from the server

Stat request:

```
     Byte/     0       |       1       |       2       |       3       |
        /              |               |               |               |
       |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
       +---------------+---------------+---------------+---------------+
      0| 0x80          | 0x10          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
      4| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
      8| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     12| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     16| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     20| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
       Total 24 bytes

   Field        (offset) (value)
   Magic        (0)    : 0x80
   Opcode       (1)    : 0x10
   Key length   (2,3)  : 0x0000
   Extra length (4)    : 0x00
   Data type    (5)    : 0x00
   Reserved     (6,7)  : 0x0000
   Total body   (8-11) : 0x00000000
   Opaque       (12-15): 0x00000000
   CAS          (16-23): 0x0000000000000000
   Extras              : None
   Key                 : None
   Value               : None
```

The server will send each value in a separate packet with an "empty" packet (no key / no value) to terminate the sequence. Each of the response packets look like the following example:

Stat response:

```
     Byte/     0       |       1       |       2       |       3       |
        /              |               |               |               |
       |0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|0 1 2 3 4 5 6 7|
       +---------------+---------------+---------------+---------------+
      0| 0x81          | 0x10          | 0x00          | 0x03          |
       +---------------+---------------+---------------+---------------+
      4| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
      8| 0x00          | 0x00          | 0x00          | 0x07          |
       +---------------+---------------+---------------+---------------+
     12| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     16| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     20| 0x00          | 0x00          | 0x00          | 0x00          |
       +---------------+---------------+---------------+---------------+
     24| 0x70 ('p')    | 0x69 ('i')    | 0x64 ('d')    | 0x33 ('3')    |
       +---------------+---------------+---------------+---------------+
     28| 0x30 ('0')    | 0x37 ('7')    | 0x38 ('8')    |
       +---------------+---------------+---------------+
       Total 31 bytes (24 byte header, 3 byte key, 4 byte body)

   Field        (offset) (value)
```

```
Magic        (0)    : 0x81
Opcode       (1)    : 0x10
Key length   (2,3)  : 0x0003
Extra length (4)    : 0x00
Data type    (5)    : 0x00
Status       (6,7)  : 0x0000
Total body   (8-11) : 0x00000007
Opaque       (12-15): 0x00000000
CAS          (16-23): 0x0000000000000000
Exstras             : None
Key                 : The textual string "pid"
Value               : The textual string "3078"
```

TOC

5. Security Considerations

Memcache has no authentication or security layers whatsoever. It is RECOMMENDED that memcache be deployed strictly on closed, protected, back-end networks within a single data center, within a single cluster of servers, or even on a single host, providing shared caching for multiple applications. Memcache MUST NOT be made available on a public network.

TOC

Appendix A. Acknowledgments

Thanks to Brad Fitzpatrick, Anatoly Vorobey, Steven Grimm, and Dustin Sallings, for their work on the memcached server.

Thanks to Sean Chittenden, Jonathan Steinert, Brian Aker, Evan Martin, Nathan Neulinger, Eric Hodel, Michael Johnson, Paul Querna, Jamie McCarthy, Philip Neustrom, Andrew O'Brien, Josh Rotenberg, Robin H. Johnson, Tim Yardley, Paolo Borelli, Eli Bingham, Jean-Francois Bustarret, Paul G, Paul Lindner, Alan Kasindorf, Chris Goffinet, Tomash Brechko, and others for their work reporting bugs and maintaining memcached client libraries and bindings in many languages.

TOC

Authors' Addresses

Aaron Stone (editor)
Six Apart, Ltd.
548 4th Street
San Francisco, CA 94107
USA
Email: aaron@serendipity.palo-alto.ca.us

Trond Norbye (editor)
Sun Microsystems, INC
Haakon VII g. 7B
Trondheim NO-7485 Trondheim
Norway
Email: trond.norbye@sun.com