YesSQL Client–Server Binary Protocol Specification

1 Overview

This document describes the binary protocol used between the YesSQL client and server. All communication consists of a binary message with the following top-level format:

Offset	Size	Description
0	8 bytes	Total message length (including these 8 bytes)
8	8 bytes	Number of elements in the response/request
16	2 bytes	Command code
18	...	Command-specific payload

All integer fields use **Big Endian** byte order.

2 Command Codes

Code	Name
0	OK
1	ERR
2	GET
3	SET
4	GET_KEYS
5	GET_KEYS_PREFIX
6	GET_FF
7	GET_FB
8	REMOVE
9	CREATE_CURSOR
10	DELETE_CURSOR
11	DATA_NOT_FOUND
12	INVALID_COMMAND

3 General String Format

Whenever a string is transmitted in the protocol, it is sent as:

- For short strings (cursor names): **1-byte length** + raw bytes
- For keys: **2-byte length** + raw bytes
- For values: **4-byte length** + raw bytes

4 SET Command

The client constructs a SET command as:

Size	Field	Description
8	Total length	Entire packet length
8	NumElems	Always 1
2	SET	Command code 3
2	KeyLen	Length of key
KeyLen	Key	UTF-8 bytes
4	ValLen	Length of value
ValLen	Value	UTF-8 bytes

5 GET Command

Size	Field	Description
8	Total length	Packet length
8	NumElems	Always 1
2	GET	Command code 2
2	KeyLen	Length of key
KeyLen	Key	UTF-8 bytes

6 REMOVE Command

Identical but with command code 8.

7 Cursor Commands

7.1 CREATE_CURSOR

Size	Field	Description
8	TotalLen	Entire packet length
8	NumElems	Always 1
2	CREATE_CURSOR	Code 9
1	CursorNameLen	Length of cursor name
NameLen	CursorName	Bytes
2	KeyLen	Key length (0 allowed)
KeyLen	Key	Starting key

7.2 DELETE_CURSOR

Size	Field	Description
8	TotalLen	Packet length
8	NumElems	Always 1
2	DELETE_CURSOR	Code 10
1	CursorNameLen	Name length
NameLen	CursorName	Name bytes

8 GET_FF / GET_FB / GET_KEYS

The structure is identical except for the command code.

Size	Field	Description
8	TotalLen	Packet length
6	Reserved	Usually zeros
2	Count	Max number of returned items
2	Command	One of 6, 7, 4
1	CursorNameLen	Length
NameLen	CursorName	Bytes

9 GET_KEYS_PREFIX

This extends the above structure with a prefix field.

Size	Field	Description
8	TotalLen	Packet length
6	Reserved	Zeros
2	Count	Maximum returned items
2	GET_KEYS_PREFIX	Code 5
1	CursorLen	Cursor name length
CursorLen	Cursor	Bytes
2	PrefixLen	Prefix length
PrefixLen	Prefix	Bytes

10 Response Format

All server responses begin with:

Size	Field	Description
8	TotalLen	Length including header
8	NumElems	Number of key/value pairs
2	CmdCode	Response type

Each returned element follows:

2 bytes	KeyLen
KeyLen	Key bytes
If values are included:	
4 bytes	ValLen
ValLen	Value bytes

Note on Error Responses

If the server returns a response with the command code **ERR** (value 1), the last two bytes of the response payload (after all standard fields) contain an **error code**. This error code is not yet implemented in the current version of the protocol, but the structure is reserved for future use.

Size	**Field**	**Description**
2 bytes	ErrorCode	Application-defined error identifier

These two bytes appear only when `CmdCode = ERR`. Clients should therefore check the command code and, if it equals `ERR`, read the final two bytes of the message as an integer error code.