Primary_Partition Server–Server Binary Protocol Specification

1 Overview

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

Offset	Size	Description
0	8 bytes	Total message length (including these 8 bytes)
8	8 bytes	Client ID (uint64_t)
16	8 bytes	Number of elements in the response/request
24	2 bytes	Command code
26	...	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

Strings are encoded as:

- Cursor names: **1-byte length** + raw bytes
- Keys: **2-byte length** + raw bytes
- Values: **4-byte length** + raw bytes

4 SET Command

SET messages follow the header and then:

Size	Field	Description
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
2	KeyLen	Length of key
KeyLen	Key	UTF-8 bytes

6 REMOVE Command

Identical to GET but with command code 8.

7 Cursor Commands

7.1 CREATE_CURSOR

Size	Field	Description
1	CursorNameLen	Length of cursor name
CursorNameLen	CursorName	Name bytes
2	KeyLen	Starting key length (0 allowed)
KeyLen	Key	Starting key bytes

7.2 DELETE_CURSOR

Size	Field	Description
1	CursorNameLen	Name length
CursorNameLen	CursorName	Bytes

8 GET_FF / GET_FB / GET_KEYS

These share a common binary layout. Please note that the first 8 bytes overlap, with the headers 8 bytes at offset 16 :

Size	Field	Description
6	Reserved	Zero-filled
2	Count	Maximum number of returned items
1	CursorNameLen	Cursor name length
CursorNameLen	CursorName	Name bytes
2	NexKeySize	size of the next key(saved in the cursor)
NextKeySize	NextKey	UTF-8 bytes

9 GET_KEYS_PREFIX

Same as above, just extended with:

Size	Field	Description
2	PrefixLen	Prefix length
PrefixLen	Prefix	UTF-8 bytes

10 Response Format

All server responses begin with the standard header:

Offset	Size	Field
0	8	TotalLen
8	8	ClientID
16	8	NumElems
24	2	CmdCode

Elements follow:

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

If the command was GET_X, then a footer is added containing:

2 bytes	NextKeyLen
NextKeyLen	NextKeyBytes
1 byte	CursorLen
CursorLen	CursorName

GET_X Partition-Jump Behavior

When executing a GET_X command, if the cursor reaches the end of a partition and must continue scanning in the next partition, the server sets a special flag inside the NumElems field.

NumElems is encoded as an 8-byte unsigned integer. The server uses the **5ths byte (offset +4 from the start of NumElems) first bit (0x01)** as a flag whenever a partition boundary is crossed.

Bitmask	**Meaning**
0x01	When set, instructs the receiver that the server jumped to a new partition and placed the **maximum key**

Only the 5th byte of NumElems is used for flags; all other bytes still encode the element count normally.

Partition_Servers must check this flag when CmdCode corresponds to GET_KEYS, GET_KEYS_PREFIX, GET_FF, or GET_FB.

Note on Error Responses

If the server returns a response with **CmdCode = ERR** (value 1), the last two bytes of the message contain a reserved **error code**. This field is not yet implemented but is reserved for future use.

Size	Field	Description
2	ErrorCode	Application-defined error identifier

Servers must read these two bytes only when `CmdCode = ERR` and they exist of course.