

QDF

Ramesh Subramonian

February 11, 2026

1 Introduction

2 QDF Data Layout

This section defines the layout of data in a QDF

2.1 qtype

The enum `qtype_t` is defined as

```
typedef enum {
    Qerr = 0,
    Q0,
    B1, // boolean stored as bit
    BL, // boolean stored as bool
    I8, // signed 1-byte integer
    I16, // signed 2-byte integer
    I32, // signed 4-byte integer
    I64, // signed 8-byte integer
    BF16, // bfloat16 floating point
    FP32, // single precision floating point
    FP64, // double precision floating point
    UI8, // unsigned 1-byte integer
    UI16, // unsigned 2-byte integer
    UI32, // unsigned 4-byte integer
    UI64, // unsigned 8-byte integer
    SC, // constant length string
    TM, // struct tm as defined in time.h
} qtype_t;
```

I32	int32_t
TM	struct tm
FP32	float
Q0	void

Table 1: Qtypes to C Types

2.2 q2c

The Lua table `q2c` is a mapping between a value of type `qtype_t` and a C type.

2.3 jtype

The enum `jtype_t` is defined as follows

```
typedef enum {
    j_error = 0,
    j_undef,
    j_nil,
    j_bool,
    j_string,
    j_number,
    j_date,
    j_array,
    j_object,
    // j_hashtable
} jtype_t;
```

2.4 QDF struct

The struct `qdf_rec_type` is defined as follows

```
typedef struct _qdf_rec_type {
    void *data;
    uint32_t size; // must be a multiple of 8
    bool is_mmap; // true => we have mmapmed data not malloc'd it
    bool is_foreign; // true => do not free() or munmap()
    bool is_read_only; // true => don't modify
} QDF_REC_TYPE;
```

Return	Name	Args	
bool	chk_qdf()	(const qdf_rec_type * const x)	
qtype_t	getq	(const qdf_rec_type * const x)	
jtype_t	get_jtype()	(const qdf_rec_type * const x)	
UI4	get_length()	(const qdf_rec_type * const x)	
const char *	get_read_arr_ptr()	(const qdf_rec_type * const x)	

Table 2: List of read only helper functions

3 Read only helper functions

Table 2 lists helper functions that have been provided to you. They take as an argument an immutable pointer to `qdf_rec_type` and do **not** modify the location pointed to.

3.1 `chk_qdf()`

Returns `true` if `x` is syntactically valid; `false`, otherwise

3.2 `get_qtype()`

Returns `qtype_t` of `x`

3.3 `get_jtype()`

Returns `jtype_t` of `x`

3.4 `get_length()`

If `jtype_t` of `x` is `j_array`, then returns length of array; else, returns 0.

3.5 `get_read_arr_ptr()`

If `jtype_t` of `x` is `j_array`, then returns `NULL`. If `qtype_t` of `x` is `Qerr` or `Q0`, then returns `NULL`. Else, returns a pointer to the 0^{th} element of the array. The data pointed to cannot be modified.

3.6 `get_num_keys()`

If `jtype_t` `x` is `j_object`, then returns number of keys; else, returns 0.