sd_bus_message_append_array

- Description
- Return Value
- See Also

systemd

sd_bus_message_append_array

3

sd_bus_message_append_array

sd_bus_message_append_array_memfd

sd_bus_message_append_array_iovec

sd_bus_message_append_array_space

Append an array of fields to a D-Bus message

#include <systemd/sd-bus.h>

int sd_bus_message_append_array

sd_bus_message *

m

char

type

const void *

ptr

size_t

size

int sd_bus_message_append_array_memfd

sd_bus_message *

m

char

type

int

memfd

uint64_t

offset

uint64_t

size

int sd_bus_message_append_array_iovec

sd_bus_message *

m

char

type

const struct iovec *

iov

unsigned

n

int sd_bus_message_append_array_space

sd_bus_message *

m

char

type

size_t

size

void **

ptr

Description

The `sd_bus_message_append_array()` function appends an array to a D-Bus message `m`. A container will be opened, the array contents appended, and the container closed. The parameter `type` determines how the pointer `p` is interpreted. `type` must be one of the "trivial" types `y`, `n`, `q`, `i`, `u`, `x`, `t`, `d` (but not `b`), as defined by the Basic D-Bus Types section of the D-Bus specification, and listed in `sd_bus_message_append_basic(3)`. Pointer `p` must point to an array of size `size` bytes containing items of the respective type. Size `size` must be a multiple of the size of the type `type`. As a special case, `p` may be `NULL`, if `size` is 0. The memory pointed to by `p` is copied into the memory area containing the message and stays in possession of the caller. The caller may hence freely change the data after this call without affecting the message the array was appended to.

The `sd_bus_message_append_array_memfd()` function appends an array of a trivial type to message `m`, similar to `sd_bus_message_append_array()`. The contents of the memory file descriptor `memfd` starting at the specified offset and of the specified size is used as the contents of the array. The offset and size must be a multiple of the size of the type `type`. However, as a special exception, if the offset is specified as zero and the size specified as UINT64_MAX the full memory file descriptor contents is used. The memory file descriptor is sealed by this call if it has not been sealed yet, and cannot be modified after this call. See `memfd_create(2)` for details about memory file descriptors. Appending arrays with memory file descriptors enables efficient zero-copy data transfer, as the memory file descriptor may be passed as-is to the destination, without copying the memory in it to the destination process. Not all protocol transports support passing memory file descriptors between participants, in which case this call will automatically fall back to copying. Also, as memory file descriptor passing is inefficient for smaller amounts of data, copying might still be enforced even where memory file descriptor passing is supported.

The `sd_bus_message_append_array_iovec()` function appends an array of a trivial type to the message `m`, similar to `sd_bus_message_append_array()`. Contents of the I/O vector array `iov` are used as the

contents of the array. The total size of `iov` payload (the sum of iov_len fields) must be a multiple of the size of the type `type`. The `iov` argument must point to `n` I/O vector structures. Each structure may have the iov_base field set, in which case the memory pointed to will be copied into the message, or unset (set to zero), in which case a block of zeros of length iov_len bytes will be inserted. The memory pointed at by `iov` may be changed after this call.

The `sd_bus_message_append_array_space()` function appends space for an array of a trivial type to message `m`. It behaves the same as `sd_bus_message_append_array()`, but instead of copying items to the message, it returns a pointer to the destination area to the caller in pointer `p`. The caller should subsequently write the array contents to this memory. Modifications to the memory pointed to should only occur until the next operation on the bus message is invoked. Most importantly, the memory should not be altered anymore when another field has been added to the message or the message has been sealed.

Return Value

On success, these calls return 0 or a positive integer. On failure, they return a negative errno-style error code.

See Also

`systemd(1)`, `sd-bus(3)`, `sd_bus_message_append(3)`, `sd_bus_message_append_basic(3)`, `memfd_create(2)`, The D-Bus specification

□□□ #2
□ □□□ □□□ 2026-10-31 14:50:09 UTC
□ □□□ □□□ 2026-10-31 14:50:10 UTC