sd_bus_message_new_method_error

- Description
- Return Value
  - Errors
- See Also

systemd

sd_bus_message_new_method_error

3

sd_bus_message_new_method_error

sd_bus_message_new_method_errorf

sd_bus_message_new_method_errno

sd_bus_message_new_method_errnof

Create an error reply for a method call

#include <systemd/sd-bus.h>

int sd_bus_message_new_method_error

sd_bus_message *

call

sd_bus_message **

m

const sd_bus_error *

e

int sd_bus_message_new_method_errorf

sd_bus_message *

call

sd_bus_message **

m

const char *

name

const char *

format

...

int sd_bus_message_new_method_errno

sd_bus_message *

call

sd_bus_message **

m

int

error

const sd_bus_error *

p

int sd_bus_message_new_method_errnof

sd_bus_message *

call

sd_bus_message **

m

int

error

const char *

format

...

Description

The `sd_bus_message_new_method_error()` function creates a new bus message object that is an error reply to the `call` message, and returns it in the `m` output parameter. The error information from error `e` is appended: the `name` field of `e` is used as the error identifier in the reply header (for example an error name such as `org.freedesktop.DBus.Error.NotSupported` or the equivalent symbolic `SD_BUS_ERROR_NOT_SUPPORTED`), and the `message` field is set as the human-readable error message string if present. The error `e` must have the `name` field set, see `sd_bus_error_is_set(3)`.

The `sd_bus_message_new_method_errorf()` function creates an error reply similarly to `sd_bus_message_new_method_error()`, but instead of a ready error structure, it takes an error identifier string `name`, plus a `printf(3)` format string `format` and corresponding arguments. An error reply is sent with the error identifier `name` and the formatted string as the message. `name` and `format` must not be `NULL`.

The `sd_bus_message_new_method_errno()` function creates an error reply similarly to `sd_bus_message_new_method_error()`, but in addition to the error structure `p`, it takes an `errno(3)` error value in parameter `error`. If the error `p` is set (see `sd_bus_error_is_set(3)`), it is used in the reply. Otherwise, `error` is translated to an error identifier and used to create a new error structure using `sd_bus_error_set_errno(3)` and that is used in the reply. (If `error` is zero, no error is actually set, and an error reply with no information is created.)

The `sd_bus_message_new_method_errnof()` function creates an error reply similarly to `sd_bus_message_new_method_error()`. It takes an `errno(3)` error value in parameter `error`, plus a `printf(3)` format string `format` and corresponding arguments. `%m` may be used in the format string to refer to the error string corresponding to the specified errno code. The error message is initialized using the error identifier generated from `error` and the formatted string. (If `error` is zero, no error is actually set, and an error reply with no information is created.)

Return Value

These functions return 0 if the error reply was successfully created, and a negative errno-style error code otherwise.

Errors

Returned errors may indicate the following problems:

`-EINVAL`
The call message `call` or the output parameter `m` are `NULL`.

Message `call` is not a method call message.

The error `e` parameter to `sd_bus_message_new_method_error()` is not set, see `sd_bus_error_is_set(3)`.

`-EPERM`
Message `call` has been sealed.

`-ENOTCONN`
The bus to which message `call` is attached is not connected.

`-ENOMEM`
Memory allocation failed.

See Also

`systemd(1)`, `sd-bus(3)`

□□□ #2
□ □□□ □□□ 2026-10-31 14:51:16 UTC
□ □□□ □□□ 2026-10-31 14:51:17 UTC