Communication app using Debus communication using C++ (sudo apt install libdbus-1-dev)

✅ server.cpp

cpp Copy Edit

```cpp
#include <dbus/dbus.h>
#include <iostream>
#include <thread>
#include <string>
#include <unistd.h>

#define OBJECT_PATH "/com/example/Chat"
#define INTERFACE_NAME "com.example.ChatInterface"
#define SERVER_NAME "com.example.Server"
#define CLIENT_NAME "com.example.Client"

// Message handler (required by D-Bus)
DBusHandlerResult message_handler(DBusConnection* conn, DBusMessage* msg, void* user_data) {
    return DBUS_HANDLER_RESULT_HANDLED;
}
```

Copy Edit

```cpp
// Message handler (required by D-Bus)
DBusHandlerResult message_handler(DBusConnection* conn, DBusMessage* msg, void* user_data) {
    return DBUS_HANDLER_RESULT_HANDLED;
}

DBusObjectPathVTable vtable = {
    .unregister_function = nullptr,
    .message_function = message_handler,
    .dbus_internal_pad1 = nullptr,
    .dbus_internal_pad2 = nullptr,
    .dbus_internal_pad3 = nullptr,
    .dbus_internal_pad4 = nullptr
};
```

```cpp
// Listen for incoming messages
void listen(DBusConnection* conn) {
    while (true) {
        dbus_connection_read_write(conn, 0);
        DBusMessage* msg = dbus_connection_pop_message(conn);
        if (msg && dbus_message_is_method_call(msg, INTERFACE_NAME, "SendMessage")) {
            DBusMessageIter args;
            if (dbus_message_iter_init(msg, &args)) {
                char* text;
                dbus_message_iter_get_basic(&args, &text);
                std::cout << "\n[client]: " << text << "\n> ";
                std::cout.flush();
            }
            DBusMessage* reply = dbus_message_new_method_return(msg);
            dbus_connection_send(conn, reply, nullptr);
            dbus_message_unref(reply);
        }
        if (msg) dbus_message_unref(msg);
        usleep(100000);
    }
}
```

```cpp
// Allow user to type and send messages
void input_loop(DBusConnection* conn) {
    std::string input;
    while (true) {
        std::cout << "> ";
        std::getline(std::cin, input);

        DBusMessage* msg = dbus_message_new_method_call(
            CLIENT_NAME, OBJECT_PATH, INTERFACE_NAME, "SendMessage"
        );
        DBusMessageIter args;
        dbus_message_iter_init_append(msg, &args);
        const char* cstr = input.c_str();
        dbus_message_iter_append_basic(&args, DBUS_TYPE_STRING, &cstr);

        DBusPendingCall* pending;
        dbus_connection_send_with_reply(conn, msg, &pending, -1);
        dbus_connection_flush(conn);
        dbus_message_unref(msg);
        dbus_pending_call_unref(pending);
    }
}
```

```cpp
int main() {
    DBusError err;
    dbus_error_init(&err);
    DBusConnection* conn = dbus_bus_get(DBUS_BUS_SESSION, &err);

    dbus_bus_request_name(conn, SERVER_NAME, DBUS_NAME_FLAG_REPLACE_EXISTING, &err);
    dbus_connection_register_object_path(conn, OBJECT_PATH, &vtable, nullptr);

    std::cout << "Server ready. Type messages below:\n";

    std::thread listener(listen, conn);
    input_loop(conn);

    listener.join();
    return 0;
}
```

✅ client.cpp

cpp Copy Edit

```cpp
#include <dbus/dbus.h>
#include <iostream>
#include <thread>
#include <string>
#include <unistd.h>

#define OBJECT_PATH "/com/example/Chat"
#define INTERFACE_NAME "com.example.ChatInterface"
#define SERVER_NAME "com.example.Server"
#define CLIENT_NAME "com.example.Client"

// Message handler (required by D-Bus)
DBusHandlerResult message_handler(DBusConnection* conn, DBusMessage* msg, void* user_data) {
    return DBUS_HANDLER_RESULT_HANDLED;
}
```

```
DBusObjectPathVTable vtable = {
    .unregister_function = nullptr,
    .message_function = message_handler,
    .dbus_internal_pad1 = nullptr,
    .dbus_internal_pad2 = nullptr,
    .dbus_internal_pad3 = nullptr,
    .dbus_internal_pad4 = nullptr
};
```

```
// Listen for incoming messages
void listen(DBusConnection* conn) {
    while (true) {
        dbus_connection_read_write(conn, 0);
        DBusMessage* msg = dbus_connection_pop_message(conn);
        if (msg && dbus_message_is_method_call(msg, INTERFACE_NAME, "SendMessage")) {
            DBusMessageIter args;
            if (dbus_message_iter_init(msg, &args)) {
                char* text;
                dbus_message_iter_get_basic(&args, &text);
                std::cout << "\n[Server]: " << text << "\n> ";
                std::cout.flush();
            }
            DBusMessage* reply = dbus_message_new_method_return(msg);
            dbus_connection_send(conn, reply, nullptr);
            dbus_message_unref(reply);
        }
        if (msg) dbus_message_unref(msg);
        usleep(100000);
    }
}
```

```
// Allow user to type and send messages
void input_loop(DBusConnection* conn) {
    std::string input;
    while (true) {
        std::cout << "> ";
        std::getline(std::cin, input);

        DBusMessage* msg = dbus_message_new_method_call(
            SERVER_NAME, OBJECT_PATH, INTERFACE_NAME, "SendMessage"
        );
        DBusMessageIter args;
        dbus_message_iter_init_append(msg, &args);
        const char* cstr = input.c_str();
        dbus_message_iter_append_basic(&args, DBUS_TYPE_STRING, &cstr);

        DBusPendingCall* pending;
        dbus_connection_send_with_reply(conn, msg, &pending, -1);
        dbus_connection_flush(conn);
        dbus_message_unref(msg);
        dbus_pending_call_unref(pending);
    }
}
```

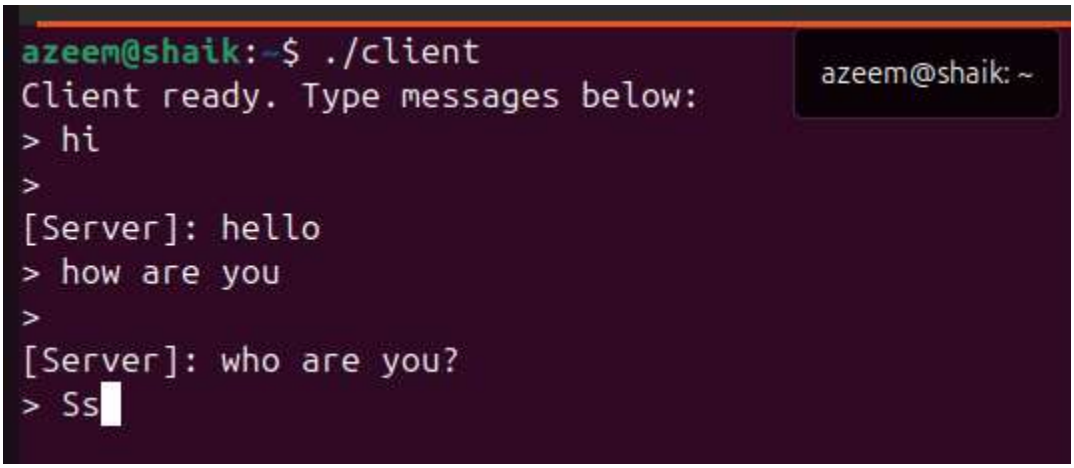
```
int main() {
    DBusError err;
    dbus_error_init(&err);
    DBusConnection* conn = dbus_bus_get(DBUS_BUS_SESSION, &err);

    dbus_bus_request_name(conn, CLIENT_NAME, DBUS_NAME_FLAG_REPLACE_EXISTING, &err);
    dbus_connection_register_object_path(conn, OBJECT_PATH, &vtable, nullptr);

    std::cout << "Client ready. Type messages below:\n";

    std::thread listener(listen, conn);
    input_loop(conn);

    listener.join();
    return 0;
}
```

🛠️ How to Compile

bash Copy Edit

```
g++ server.cpp -o server `pkg-config --cflags --libs dbus-1`
g++ client.cpp -o client `pkg-config --cflags --libs dbus-1`
```

Client out put



Server out put