How to Use DispatchMessage in Windows Applications

DispatchMessage is a crucial function in the Windows API used for message handling in Windows applications. It is part of the message loop that processes messages sent to a window. Understanding how to use DispatchMessage is essential for developers creating GUI applications in Windows, as it ensures that user inputs and other system messages are appropriately handled.

In the context of a Windows application, the message loop retrieves messages from the application's message queue and dispatches them to the appropriate window procedures. The DispatchMessage function is responsible for sending these messages to the window procedure associated with the window that is the target of the message.

Examples:

1. **Basic Message Loop with DispatchMessage:**

```
#include <windows.h>

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lP
aram) {
    switch (uMsg) {
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
        default:
            return DefWindowProc(hwnd, uMsg, wParam, lParam);
    }
}

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pC
mdLine, int nCmdShow) {
    const wchar_t CLASS_NAME[] = L"Sample Window Class";

    WNDCLASS wc = {};
    wc.lpfnWndProc = WindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = CLASS_NAME;

    RegisterClass(&wc);

    HWND hwnd = CreateWindowEx(
        0,
        CLASS_NAME,
        L"Learn DispatchMessage",
```

```
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance,
        NULL
    );

    if (hwnd == NULL) {
        return 0;
    }

    ShowWindow(hwnd, nCmdShow);

    MSG msg = {};
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}
```

This example demonstrates a basic Windows application with a message loop that uses DispatchMessage to send messages to the window procedure.

2. **Handling Custom Messages:**

```
#include <windows.h>

#define WM_CUSTOM_MESSAGE (WM_USER + 1)

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lP
aram) {
    switch (uMsg) {
        case WM_CUSTOM_MESSAGE:
            MessageBox(hwnd, L"Custom Message Received!", L"Info", MB_OK);
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
        default:
            return DefWindowProc(hwnd, uMsg, wParam, lParam);
    }
}

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pC
mdLine, int nCmdShow) {
```

```
    const wchar_t CLASS_NAME[] = L"Sample Window Class";

    WNDCLASS wc = {};
    wc.lpfnWndProc = WindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = CLASS_NAME;

    RegisterClass(&wc);

    HWND hwnd = CreateWindowEx(
        0,
        CLASS_NAME,
        L"Learn DispatchMessage",
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance,
        NULL
    );

    if (hwnd == NULL) {
        return 0;
    }

    ShowWindow(hwnd, nCmdShow);

    // Post a custom message
    PostMessage(hwnd, WM_CUSTOM_MESSAGE, 0, 0);

    MSG msg = {};
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}
```

In this example, a custom message (WM_CUSTOM_MESSAGE) is defined and handled in the window procedure. The message is posted using PostMessage, and DispatchMessage ensures it is sent to the appropriate window procedure.