Using Raw Input

Article • 03/29/2021

This section includes sample code for the following purposes:

Registering for Raw Input

Example 1

In this sample, an application specifies the raw input from game controllers (both game pads and joysticks) and all devices off the telephony usage page except answering machines.

C++

```
RAWINPUTDEVICE Rid[4];

Rid[0].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[0].usUsage = 0x05;              // HID_USAGE_GENERIC_GAMEPAD
Rid[0].dwFlags = 0;                 // adds game pad
Rid[0].hwndTarget = 0;

Rid[1].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[1].usUsage = 0x04;              // HID_USAGE_GENERIC_JOYSTICK
Rid[1].dwFlags = 0;                 // adds joystick
Rid[1].hwndTarget = 0;

Rid[2].usUsagePage = 0x0B;          // HID_USAGE_PAGE_TELEPHONY
Rid[2].usUsage = 0x00;
Rid[2].dwFlags = RIDEV_PAGEONLY;    // adds all devices from telephony
page
Rid[2].hwndTarget = 0;

Rid[3].usUsagePage = 0x0B;          // HID_USAGE_PAGE_TELEPHONY
Rid[3].usUsage = 0x02;              //
HID_USAGE_TELEPHONY_ANSWERING_MACHINE
Rid[3].dwFlags = RIDEV_EXCLUDE;     // excludes answering machines
Rid[3].hwndTarget = 0;

if (RegisterRawInputDevices(Rid, 4, sizeof(Rid[0])) == FALSE)
```

```
{
    //registration failed. Call GetLastError for the cause of the error.
}
```

Example 2

In this sample, an application wants raw input from the keyboard and mouse but wants to ignore legacy keyboard and mouse window messages (which would come from the same keyboard and mouse).

C++

```
RAWINPUTDEVICE Rid[2];

Rid[0].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[0].usUsage = 0x02;              // HID_USAGE_GENERIC_MOUSE
Rid[0].dwFlags = RIDEV_NOLEGACY;    // adds mouse and also ignores legacy
mouse messages
Rid[0].hwndTarget = 0;

Rid[1].usUsagePage = 0x01;          // HID_USAGE_PAGE_GENERIC
Rid[1].usUsage = 0x06;              // HID_USAGE_GENERIC_KEYBOARD
Rid[1].dwFlags = RIDEV_NOLEGACY;    // adds keyboard and also ignores
legacy keyboard messages
Rid[1].hwndTarget = 0;

if (RegisterRawInputDevices(Rid, 2, sizeof(Rid[0])) == FALSE)
{
    //registration failed. Call GetLastError for the cause of the error
}
```

Performing a Standard Read of Raw Input

This sample shows how an application does an unbuffered (or standard) read of raw input from either a keyboard or mouse Human Interface Device (HID) and then prints out various information from the device.

C++

```
case WM_INPUT:
{
    UINT dwSize;

    GetRawInputData((HRAWINPUT)lParam, RID_INPUT, NULL, &dwSize,
sizeof(RAWINPUTHEADER));
    LPBYTE lpb = new BYTE[dwSize];
    if (lpb == NULL)
```

```
    {
        return 0;
    }

    if (GetRawInputData((HRAWINPUT)lParam, RID_INPUT, lpb, &dwSize,
sizeof(RAWINPUTHEADER)) != dwSize)
         OutputDebugString (TEXT("GetRawInputData does not return correct
size !\n"));

    RAWINPUT* raw = (RAWINPUT*)lpb;

    if (raw->header.dwType == RIM_TYPEKEYBOARD)
    {
        hResult = StringCchPrintf(szTempOutput, STRSAFE_MAX_CCH,
            TEXT(" Kbd: make=%04x Flags:%04x Reserved:%04x
ExtraInformation:%08x, msg=%04x VK=%04x \n"),
            raw->data.keyboard.MakeCode,
            raw->data.keyboard.Flags,
            raw->data.keyboard.Reserved,
            raw->data.keyboard.ExtraInformation,
            raw->data.keyboard.Message,
            raw->data.keyboard.VKey);
        if (FAILED(hResult))
        {
        // TODO: write error handler
        }
        OutputDebugString(szTempOutput);
    }
    else if (raw->header.dwType == RIM_TYPEMOUSE)
    {
        hResult = StringCchPrintf(szTempOutput, STRSAFE_MAX_CCH,
            TEXT("Mouse: usFlags=%04x ulButtons=%04x usButtonFlags=%04x
usButtonData=%04x ulRawButtons=%04x lLastX=%04x lLastY=%04x
ulExtraInformation=%04x\r\n"),
            raw->data.mouse.usFlags,
            raw->data.mouse.ulButtons,
            raw->data.mouse.usButtonFlags,
            raw->data.mouse.usButtonData,
            raw->data.mouse.ulRawButtons,
            raw->data.mouse.lLastX,
            raw->data.mouse.lLastY,
            raw->data.mouse.ulExtraInformation);

        if (FAILED(hResult))
        {
        // TODO: write error handler
        }
        OutputDebugString(szTempOutput);
    }

    delete[] lpb;
    return 0;
}
```

Performing a Buffered Read of Raw Input

This sample shows how an application does a buffered read of raw input from a generic HID.

C++

```
case MSG_GETRIBUFFER: // Private message
    {
    UINT cbSize;
    Sleep(1000);

    VERIFY(GetRawInputBuffer(NULL, &cbSize, sizeof(RAWINPUTHEADER)) ==
0);
    cbSize *= 16; // up to 16 messages
    Log(_T("Allocating %d bytes"), cbSize);
    PRAWINPUT pRawInput = (PRAWINPUT)malloc(cbSize);
    if (pRawInput == NULL)
    {
        Log(_T("Not enough memory"));
        return;
    }

    for (;;)
    {
        UINT cbSizeT = cbSize;
        UINT nInput = GetRawInputBuffer(pRawInput, &cbSizeT,
sizeof(RAWINPUTHEADER));
        Log(_T("nInput = %d"), nInput);
        if (nInput == 0)
        {
            break;
        }

        ASSERT(nInput > 0);
        PRAWINPUT* paRawInput = (PRAWINPUT*)malloc(sizeof(PRAWINPUT) *
nInput);
        if (paRawInput == NULL)
        {
            Log(_T("paRawInput NULL"));
            break;
        }

        PRAWINPUT pri = pRawInput;
        for (UINT i = 0; i < nInput; ++i)
        {
            Log(_T(" input[%d] = @%p"), i, pri);
            paRawInput[i] = pri;
            pri = NEXTRAWINPUTBLOCK(pri);
        }

        free(paRawInput);
    }
```

```
    free(pRawInput);
}
```

Feedback

Was this page helpful? Yes No